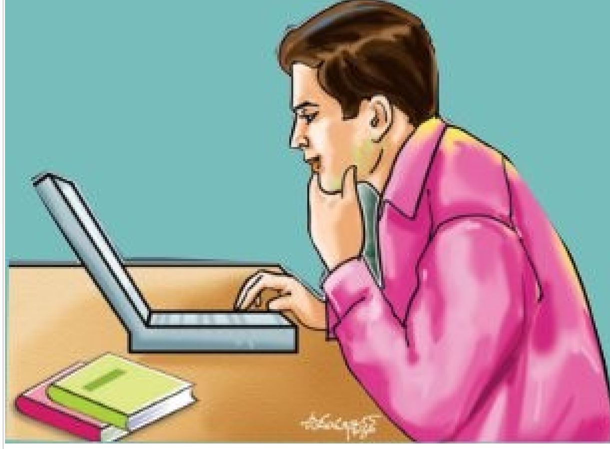
http://www.eenadupratibha.net/Engineering-Colleges/Engineering-Jobs/c_ds_clanguage-tel.html

సులభంగా 'సీ' పాఠాలు..!

ప్రారంభంలో కంప్యూటర్‌కు సూచనలు ఇవ్వడానికి, మెషిన్ లాంగ్వేజీని ఉపయోగించేవారు. ఈ మెషిన్ లాంగ్వేజీలో ఏ విషయాన్నయినా చెప్పడానికి జీరోలు, ఒకట్లలో తెలపాల్సి ఉంటుంది. ఇది మనకు కొంచెం కష్టమైన పనే.



- అందువల్ల 1950 లలో మనం వాడే భాష(ఇంగ్లిష్)కు దగ్గరగా ఉండే లాంగ్వేజీలను అభివృద్ధి చేయడంపై ప్రధానంగా దృష్టి సారించారు. ఈవిధమైన ప్రోగ్రామింగ్ లాంగ్వేజీలను High Level లాంగ్వేజీలు అంటారు. మనం ఉపయోగించే భాషకు దగ్గరగా ఉండే భాషలకు చెందినవారికి కూడా ఇవి సులభంగా అర్థమవుతాయి. అంతేకాకుండా, మెషిన్ లాంగ్వేజీలు కంప్యూటరు, కంప్యూటరుకూ మారుతూ ఉంటాయి. అదే High Level లాంగ్వేజీలు ఏ కంప్యూటరులోనైనా పనిచేస్తాయి. దీన్నే "Portability" అంటారు. ఇది ఒక ప్రోగ్రామింగ్ లాంగ్వేజ్ యొక్క మంచి లక్షణం. 1950 లో IBM కంపెనీ FORTRAN అనే High Level లాంగ్వేజీని అభివృద్ధి చేసింది. ఇప్పటికీ దీన్ని సైంటిఫిక్ కంప్యూటింగ్ (వాతావరణ పరిస్థితులను అంచనా వేయడం, ఉపగ్రహాలను కక్ష్యలో ప్రవేశపెట్టే విధానం)లో ఉపయోగిస్తున్నారు.

AT&T Bell ల్యాబ్స్‌కు చెందిన కెన్ థాంప్సన్, డెన్నిస్ రిచీ 1969-73 మధ్య 'సీ' లాంగ్వేజీని అభివృద్ధి చేశారు. అంతకుముందు అభివృద్ధి చేసిన లాంగ్వేజీలను B, BCPL అని పిలవడం వల్ల ఈ కొత్త లాంగ్వేజీకి సీ అని పేరు పెట్టారు. ఎలాంటి HW Architecture ఉండే కంప్యూటరులోనైనా పనిచేయగలగడం FORTRAN లాంటి లాంగ్వేజీలతో పోలిస్తే, సీ లాంగ్వేజీకి ఉన్న ప్రత్యేక లక్షణం.

సాఫ్ట్‌వేర్లను ప్రధానంగా రెండు రకాలుగా విభజించవచ్చు. అవి:

1) సిస్టమ్ సాఫ్ట్‌వేర్ 2) అప్లికేషన్ సాఫ్ట్‌వేర్.

ఆపరేటింగ్ సిస్టం, నెట్‌వర్క్ డ్రైవర్స్, డివైస్ డ్రైవర్స్ మొదలైనవాటిని సిస్టమ్ సాఫ్ట్‌వేర్‌లు అంటారు. బ్యాంకింగ్ సాఫ్ట్‌వేర్, గేమ్స్, రైల్వే రిజర్వేషన్ సాఫ్ట్‌వేర్‌లు, సెల్‌ఫోన్‌లో ఉపయోగించే సాఫ్ట్‌వేర్లను అప్లికేషను సాఫ్ట్‌వేర్‌లు అంటారు. సీ లాంగ్వేజీని ముందు రోజుల్లో సిస్టమ్ సాఫ్ట్‌వేర్‌లను అభివృద్ధి చేయడానికి ఉపయోగించేవారు. ఉదాహరణకు UNIX ఆపరేటర్ సిస్టంను 'సీ' లాంగ్వేజీలో అభివృద్ధి చేశారు. మొదటి Microsoft windows వెర్షన్లను కూడా 'సీ'లోనే అభివృద్ధి చేశారు. అంతేకాకుండా Oracle, ఇంజిన్, జావా కంపైలర్ల లాంటివాటిని కూడా సీ లాంగ్వేజీలోనే అభివృద్ధి చేశారు.

అసలు లాంగ్వేజీ ఎందుకు?

ఏ భాషనైనా ప్రధానంగా భావ వ్యక్తీకరణ కోసమే ఉపయోగిస్తాం. మనం సరైన పద్ధతిలో వ్యాకరణాన్ని ఉపయోగించి మాట్లాడితే ఎవరికైనా ఒకేవిధంగా అర్థమవుతుంది. సినిమాల్లో కమెడియన్లు చెప్పే డైలాగులను

గమనించండి. వాటిని ఒక్కొక్కరు ఒక్కోవిధంగా అర్థం చేసుకుంటారు. ఎవరికి అర్థమైనవిధంగా వారు నవ్వుకుంటారు. పదాలకు ఒకటి కంటే ఎక్కువ అర్థాలు ఉండటం కూడా దీనికి కారణం. ఇదేమాదిరిగా ఒక వాక్యానికి రెండు మూడు అర్థాలు ఉండేట్లు కంప్యూటరుకు చెబితే, అది ఏ అర్థాన్ని తీసుకొని పని చేయాలి? ఇలాంటి సమస్యలు లేకుండా High Level లాంగ్వేజీల్లో Strict గ్రామర్ ఉంటుంది. ఇది సీ లాంగ్వేజీకీ వర్తిస్తుంది.
అంతేకాకుండా 'సీ'కి Low Level లాంగ్వేజీలకు ఉండే లక్షణాలు కూడా ఉన్నాయి. Direct మెషిన్ పార్ట్‌లను Access చేయడం 'సీ'లోనూ సాధ్యమవుతుంది. అందువల్ల 'సీ'ని 'Medium Level language' అని కూడా అంటారు. 1983 లో అమెరికన్ నేషనల్ స్టాండర్డ్స్ ఇన్‌స్టిట్యూట్ (ANSI) సీ లాంగ్వేజీకి స్టాండర్డ్స్ నిర్ణయించింది. 'సీ'ని ANSIC లాంగ్వేజీ అని కూడా అంటారు.
ఒక భాషలో పట్టుసాధిస్తే దాన్ని ఉపయోగించి పొగడవచ్చు లేదా తిట్టవచ్చు. అదేవిధంగా సీ లాంగ్వేజీని కూడా మంచి పనులకు ఉపయోగిస్తున్నారు, వైరస్‌లను అభివృద్ధి చేయడానికీ వాడుతున్నారు. సీ లాంగ్వేజ్ లేకుంటే వైరస్ సాఫ్ట్‌వేర్లను సృష్టించడమే సాధ్యమయ్యేది కాదంటే అతిశయోక్తి కాదు.
సీ లాంగ్వేజీ నుంచే C++, java, Object C అనే లాంగ్వేజీలు వృద్ధిలోకి వచ్చాయి. సీ తెలిస్తే, మిగిలినవాటిని సులభంగా నేర్చుకోవచ్చు. ప్రపంచవ్యాప్తంగా Gaming Engines లో ఎక్కువ భాగం C++ నే వాడుతున్నారు. Internet, మొబైల్ ప్రోగ్రాములలో జావాను ఎక్కువగా ఉపయోగిస్తున్నారు. ఉదాహరణకు బ్లాక్‌బెర్రీ ఫోనుకు సంబంధించిన ఆప్స్‌ను Object C లో అభివృద్ధి చేస్తున్నారు. ఈవిధంగా సీ లాంగ్వేజీని గత 45 సంవత్సరాలుగా వివిధ అప్లికేషన్లలో ఉపయోగిస్తున్నారు. సీ ప్రాధాన్యాన్ని గుర్తించి, అన్ని రకాల డిగ్రీ, ఇంజినీరింగ్, సీఏ కోర్సుల్లో దీన్ని తప్పనిసరి సబ్జెక్టుగా ప్రవేశపెట్టారు. అంతేకాకుండా ఎంబెడెడ్ సిస్టమ్స్, సిస్టమ్ ప్రోగ్రామింగ్, టెలికమ్యూనికేషన్ రంగాల్లో సీ లాంగ్వేజీ తప్పనిసరి. క్యాంపస్ ఇంటర్వ్యూల్లోనూ సీ లాంగ్వేజీపైనే ఎక్కువగా ప్రశ్నలు అడుగుతున్నారు. సీ లాంగ్వేజీపై పట్టులేకపోవడం వల్ల చాలామంది అభ్యర్థులు ఉద్యోగావకాశాలను అందిపుచ్చుకోవడంలేదు. వీటన్నింటినీ దృష్టిలో ఉంచుకొని అన్నిరకాల అభ్యర్థులకు తేలిగ్గా అర్థమయ్యేరీతిలో సీ పాఠాలను అందిస్తున్నాం..

మొదటి 'సీ' ప్రోగ్రాం రాద్దాం ...రండి!



ఈ రోజుల్లో ఆండ్రాయిడ్ (Android) , విండోస్ (Windows) అనే పేర్లు తెలియనివారుండరు. వీటన్నింటినీ ఆపరేటింగ్ సిస్టమ్స్ అంటారు. వీటిని ఉపయోగించి హార్డ్‌వేర్ ద్వారా మనకు కావాల్సిన పనులన్నింటినీ చేస్తాం. ఈ ఆపరేటింగ్ సిస్టమ్స్‌ను అందరికీ సాధారణంగా (Common) అవసరమైన పనులు చేసేవిధంగా రూపొందిస్తారు.

- కొన్నిసార్లు మనకు కొత్త అవసరాలు ఏర్పడతాయి. అలాంటప్పుడు మనకు కావాల్సిన పని కోసం ఆపరేటింగ్ సిస్టమ్స్‌ను అప్‌డేట్ చేయాలి. దీనికోసమే ప్రోగ్రామింగ్ చేయాల్సి వస్తుంది. ప్రోగ్రామింగ్ చేయడానికి C, C++, java లాంటి కంప్యూటర్ భాషలను ఉపయోగిస్తాము. ఈ లాంగ్వేజీలన్నీ ఆంగ్లభాషకు దగ్గరగా ఉంటాయి. కాబట్టి మనకేం కావాలో సులభంగానే చెప్పవచ్చు. కానీ కంప్యూటరు బైనరీ సిస్టం (సున్నాలు, ఒకట్లు)పై పనిచేస్తుంది. దీన్ని 'మెషిన్ లాంగ్వేజీ' అంటారు. కాబట్టి మనం పైన పేర్కొన్న లాంగ్వేజీలలో ప్రోగ్రామ్ రాసిన తర్వాత మెషిన్ లాంగ్వేజీలోకి మార్చాల్సి ఉంటుంది. దీనికోసం కంపైలర్ (Compiler) అనే సాఫ్ట్‌వేర్‌ను ఉపయోగిస్తారు. C లాంగ్వేజ్‌కు, java కు వేర్వేరు కంపైలర్లు ఉంటాయి. ఇప్పుడు C లాంగ్వేజీ గురించి నేర్చుకుందాం...
- సాధారణంగా ప్రతి లాంగ్వేజీకి కొన్ని నియమాలు ఉంటాయి. ఒక వాక్యంలో కర్త, కర్మ, క్రియల మాదిరిగా ఒక్కో నియమానికి ఒక్కో ప్రాధాన్యం ఉంటుంది. లాంగ్వేజీ నేర్చుకోవడం అంటే దానికి సంబంధించిన నియమాలను తెలుసుకోవడమే. ప్రోగ్రామింగ్ లాంగ్వేజ్ దికి కూడా ఇదే నియమం వర్తిస్తుంది. ఉదాహరణకు ఒక ఉత్తరం రాసేటప్పుడు మనం To address, From address, Yous sincerely లాంటివి వాడినట్లే, C ప్రోగ్రామ్‌లో మూడు ముఖ్యమైనవాటిని ఉపయోగిస్తాం. అవి:
  1) #include బ్లాక్
  2) main బ్లాక్
  3) function బ్లాక్
  ముందుగా Function బ్లాక్ లేకుండా ఉండే చిన్న ది ప్రోగ్రాములను ఎలా రాయాలో తెలుసుకుందాం.
  గత 3 దశాబ్దాలుగా చాలా మంది C ప్రోగ్రాములను తయారుచేశారు. మనం వాటిని ఉపయోగించుకొని మన ప్రోగ్రామును రాయాలనుకుంటే, ఆ విషయాన్ని కంపైలరుకు తెలియజేయడానికి కొన్ని లైన్లను రాస్తాం. ఇలాంటివాటిని include బ్లాక్‌లో రాయాలి. ఉదాహరణకు Keyboard నుంచి డేటాను తీసుకోవడానికి, Screen పై విషయాలను రాయడానికి Ready Made Programms అనేకం ఉన్నాయి. వాటిని వాడుకోవడానికి కింది పేర్కొన్నవిధంగా రాయాలి.
  # include <stdio.h>
  అదేవిధంగా, sin, cos, logrithm విలువలను కనుక్కోవడానికి కూడా ready made ప్రోగ్రాములు చాలానే ఉన్నాయి. వాటిని మన ప్రోగ్రామ్‌లో వాడుకోవడానికి # include <math.h> అని రాయాలి.

main బ్లాక్ ప్రోగ్రామును కిందివిధంగా రాస్తాం.

```
#include <stdio.h>
# include <math.h>        } include బ్లాక్
int main()
{
-
- మన ప్రోగ్రామ్ main బ్లాక్      main బ్లాక్
-
-
return (0);
}
```

ఈ main program నే డ్రైవర్ అంటారు. మన ప్రోగ్రామ్ main నుంచి start అవుతుంది. అందువల్ల దీన్ని ప్రోగ్రామ్‌కు Entry Point అంటారు.

ఇప్పుడు run చేస్తే కిందివిధంగా output వచ్చేలా చిన్న సీ ప్రోగ్రామును రాద్దాం.. కావాల్సిన

Output:

Hello Welcome to EENADU Pratibha
Hello
Welcome
to
EENADU
Pratibha
Hello Welcome to EENADU Pratibha

దీని కోసం printf అనే readymade ప్రోగ్రామును ఉపయోగించాలి. దీన్ని ఉపయోగించుకోవడానికి ముందుగా include <stdio.h> రాయాలి. Screen పై ఏదైనా మెసేజ్‌ను రాయాలని printf కు చెప్పాలంటే ఆ మెసేజ్‌ను రెండు double quotes మధ్యలో ఉంచాలి. అదేవిధంగా, వాటిలోపల \t వాడితే Tab గా పనిచేస్తుంది. Next లైనుకి వెళ్లాలంటే \n ను ఉపయోగించాలి.

సీ ప్రోగ్రామ్.

```
#include <stdio.h>
int main()
{
printf ("Hello Welcome to EENADU Pratibha \n");
printf ("Hello \n Welcome \n to \n EENADU \n Pratibha \n");
printf ("Hello \t Welcome \t to \t EENADU \t Pratibha\t ");
return (0);
}
```

Output:

```
Turbo C++ IDE
≡ File Edit Search Run Compile Debug Project
EENADU1.CPP
#include<stdio.h>

int main()
{
printf("Hello Welcom to EENADU Pratibha \n");
printf("Hello \n Welcom \n to \n EENADU \n Pratibha \n");
printf("Hello \t Welcom \t to \t EENADU \t Pratibha \t");

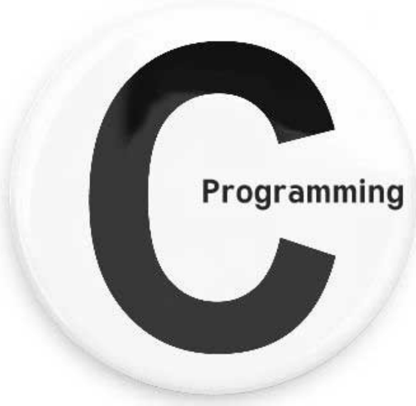
return 0;
}
5:1
```

ఈ ప్రోగ్రామును run చేయడానికి Turbo C/ C++ ను ఎలా వాడాలో తెలుసుకుందాం.Windows desktop పై ఉండే Turbo C/ C++ ఐకాన్‌ను నొక్కితే ఈ సాఫ్ట్‌వేర్ windows వస్తుంది. దీంట్లో కింద సూచించిన కమాండ్లను ఉపయోగించి ప్రోగ్రామింగ్ చేయవచ్చు.

* Compile చేసి run చేయడానికి "Ctr+F9"
* Compile చేయడానికి "Alt+F9"
* Save చేయడానికి "F2"
* ఉన్న ప్రోగ్రామును Load చేయడానికి "F3"
* ప్రోగ్రామ్ నుంచి బయటకు రావడానికి "Alt+X"
* Result స్క్రీన్‌కు వెళ్లడానికి "Alt+F5"

తర్వాతి పాఠాల్లో ప్రాక్టికల్ ప్రోగ్రాములను ఎలా రాయాలో తెలుసుకుందాం...

ఫలితాన్ని రాబట్టే పదాలివే..!

	'సీ' లాంగ్వేజ్‌లో కీ బోర్డు నుంచి సంకేతాలను తీసుకొని, వాటితో కాలిక్యులేషన్స్ చేసి, ఫలితాలను స్క్రీన్‌పై display చేసే ప్రోగ్రామింగ్ చాలా ముఖ్యమైంది. ఈ ప్రోగ్రాం రాయడానికి నాలుగు ముఖ్యమైన విషయాలను గుర్తుంచుకోవాలి అవి: a) Variable declaration

- b) Data reading
  c) Calculations
  d) Printing Results
- **1) Variables declare చేయడం**
  ప్రోగ్రాం రన్ అవుతున్నప్పుడు RAM ను కంప్యూటరు వాడుకుంటుంది. మనం ప్రోగ్రాం రాసేటప్పుడు ఇచ్చిన సంఖ్యలతోపాటు కాలిక్యులేట్ చేయగా వచ్చిన సంఖ్యలను కూడా నిక్షిప్తం (Store) చేయాల్సి ఉంటుంది. వీటికి మెమరీ కేటాయించడానికి (allocate) variables declare చేస్తాం. ఉదాహరణకు, గణితంలో మనం కనుక్కోవాల్సిన దాన్ని x అనుకొని సాధిస్తాం. ఇక్కడ x ని variable అంటాం. అలాగే programming లో కూడా variables ను వాడతాం. ఇలా variables ని declare చేయడం అనేది అన్ని programing లాంగ్వేజీల్లోనూ కీలకమైన అంశం.

సీ లాంగ్వేజీలో తీసుకునే variable names లో Upper case, Lower case, 0 - 9, Underscore (_) అక్షరాలను మాత్రమే వాడతాం. ఈ విషయంలో ఒక్కో programming లాంగ్వేజీకి ఒక్కోవిధంగా నియమాలు ఉంటాయి. సీ లాంగ్వేజీలో తీసుకునే variable లో మొదటి అక్షరం Upper Case, Lower Case, Underscore లలో ఏదో ఒకదాన్ని మాత్రమే ఉపయోగించాలి.

- అంతేకాకుండా మనం తీసుకునే variable names, C లాంగ్వేజీలో ఉండే reserved words కాకూడదు. ఉదాహరణకు ప్రోగ్రాంలో తీసుకునే variable పేరు Main, include లుగా ఉండకూడదు. అంతేకాకుండా ఈ variable ను ఒక్కసారి మాత్రమే declare చేయాలి (ఒకసారి డిక్లేర్ చేసిన variable ని ప్రోగ్రాంలో ఎన్నిసార్లయినా వాడుకోవచ్చు). అన్ని variables నూ, block

designing లో మాత్రమే declare చేయాలి. అంటే, Opening Curly braces ({) తర్వాత మాత్రమే declare చేయాలి.

- ఉదాహరణ: ప్రోగ్రాంలో వాడాల్సిన Variable Names Salary, Profit, My_salary, Total_Income

ప్రోగ్రాంలో వాడకూడని variable Names
Rama.rao (full stop ఉండకూడదు)
my-salary (minus ఉండకూడదు)

Int (reserved variable):

చాలా సీ లాంగ్వేజ్ కంపైలర్లలో, తీసుకొనే variable పేరు పదకొండు (11) అక్షరాల కంటే ఎక్కువగా ఉండకూడదు. కానీ ఇటీవల వస్తున్న సీ కంపైలర్లు పెద్ద variable పేర్లను కూడా support చేస్తున్నాయి. ఒక variable ను డిక్లేర్ చేసినప్పుడు అందులో ఎలాంటి సంఖ్యలను నిక్షిప్తం చేయాలనుకుంటున్నామో కూడా తెలపాలి. దీన్నే variable కు type ను ఇవ్వడం అంటాం. ఏదైనా variable లో పూర్ణ సంఖ్యలను store చేయాలంటే int type అని, పెద్ద పూర్ణ సంఖ్యలను store చేయాలంటే float type అని డిక్లేర్ చేయాలి. అలా కాకుండా వాస్తవ సంఖ్యలను (రియల్ నెంబర్స్) స్టోర్ చేయాలంటే float type అని డిక్లేర్ చేయాలి. సాధారణంగా float type అని రాస్తే కాలిక్యులేషన్‌లో 6th decimal వరకూ ఉపయోగించుకోగలుగుతాం. ఇంకా accurate గా కావాలంటే, variable ను double type అని డిక్లేర్ చేయాలి. ఈ సందర్భంలో కాలిక్యులేషన్స్ పదకొండు డెసిమల్ స్థానాల వరకూ వస్తాయి. ఒకవేళ variable లో అక్షరాలను store చేయాలంటే, ఆ variable ను character type అని declare చేయాలి.

ప్రోగ్రాంలో memory allocate చేయాలంటే రెండు విషయాలు గుర్తుంచుకోవాలి.
a. variable name
b. variable type

ఉదాహరణ: ముగ్గురు విద్యార్థులు పదోతరగతిలో సాధించిన మార్కులను నిక్షిప్తం చేయాలంటే మూడు variables ను int type గా డిక్లేర్ చేయాలి. ఎందుకంటే మార్కులు పూర్ణ సంఖ్యలు.

ఈ మూడు variables ను కిందివిధంగా డిక్లేర్ చేయాలి.

```
int a, b, c;
```

ఈ statement ను variable declaration statement అని అంటారు. లైన్ చివరన ';' (సెమీకొలన్)ను ఉంచడాన్ని మరిచిపోవద్దు.

సీ లాంగ్వేజ్‌లో ప్రతి Executable Statement (మెషిన్ లాంగ్వేజ్‌లోకి translate అయ్యేది)కు చివరన ';' (సెమీకొలన్) ఉండాలి. కంప్యూటరు పైన చెప్పిన statements ని చూసినప్పుడు, మూడు పూర్ణ సంఖ్యలను store (నిక్షిప్తం) చేయడానికి ఎంత మెమరీ కావాలో అంతే కేటాయిస్తుంది.

అదేవిధంగా ఒక గది పొడవు, వెడల్పు, ఎత్తును నిక్షిప్తం చేయాలంటే మూడు float type variables అవసరం. ఎందుకంటే, ఈ కొలతలు ఎప్పుడూ

పూర్ణ సంఖ్యలుగా ఉండకపోవచ్చు. float type variables ను, కిందివిధంగా రాస్తాం.

float L, B, H;

or

float Length, Breadth, Height;

or

Float My_Room_Length, My_Room_Breadth, My_Room_Height;

ఏ type variable కు ఎంత మెమరీ allocate అవుతుందో తెలుసుకుందాం.

* Char 1 byte
* int 2 or 4 bytes
* long 4 bytes
* float 4 bytes
* double 8 bytes

ఒకసారి మొమరీ కేటాయించిన తర్వాత, ఆ variable memory లో ఏముంటుందన్న విషయాన్ని ఎవరూ చెప్పలేరు. మొమరీ కేటాయించిన variables ను Un-initialised variables అంటారు. వాటి values ను garbage values అంటారు. లేదా God Only Knows Values అని కూడా అంటారు.

Un-initialised variables ను కాలిక్యులేషన్‌లలో ఎప్పుడూ వాడకూడదు. ఒకవేళ వాడినా ఫలితం garbage గానే వస్తుంది. ఒక Variable లోకి meanigful value రీడింగ్ ద్వారా రావచ్చు, direct assignment (int a = 10;) ద్వారా రావచ్చు లేదా కాలిక్యులేషన్స్ ద్వారా రావచ్చు. మనం un-intialised variable ను కాలిక్యులేషన్స్‌లో వాడితే, java లాంటి modern languages (5th generation languages)లో errors వస్తాయి. కానీ 'సీ'లో ఇలా రావు. అందువల్లే Java ను strongly typed language అని 'సీ'ని weakly typed language అని అంటారు.

2) డేటా Read చేయడం

డేటా రీడ్ చేసిన తర్వాత వాటిని నిక్షిప్తం చేయడానికి ముందే varaible ను declare చేయడం ద్వారా వాటికి మెమరీ కేటాయిస్తాం. ఇప్పుడు ఆ మెమరీలోకి కీబోర్డు ద్వారా data values ఎలా రీడ్ చేయాలో తెలుసుకుందాం. దీని కోసం scanf అనే ready made function ను (printf లా) ఉపయోగించాలి. దీన్ని వాడుకోవాలంటే, ముందుగా #include <stdio.h> అనే statement ను program లో రాయాలి. ప్రోగ్రాం రాసేటప్పుడ scanf function కు రెండు విషయాలు డిక్లేర్ చేయాలి. అవి:

a. కీ బోర్డు నుంచి ఎన్ని values, ఎలాంటి values రీడ్ చేయాలి. ఈ విషయాన్ని చెప్పడానికి, format string ను ఉపయోగిస్తాం.

ఉదాహరణ: కీబోర్డు నుంచి మూడు పూర్ణసంఖ్యలను రీడ్ చేయమని చెప్పడానికి, మనం

"%d%d%d" ను scanf కు మొదటి argument గా ఇస్తాం. అదే నాలుగు పూర్ణ సంఖ్యలు, రెండు వాస్తవ సంఖ్యలను రీడ్ చేయమని చెప్పడానికి "%d%d%d %d%f%f" అని రాస్తాం. ఇక్కడ %d అంటే పూర్ణసంఖ్య అని, %f అంటే వాస్తవ సంఖ్య అని Scanf అర్థం చేసుకుంటుంది. అలాగే, %ld, %df, %c లను long, double, character type లకు ఉపయోగిస్తాం.

b. రీడ్ చేసిన విలువలను RAM లో ఎక్కడ పెట్టాలో కూడా scanf కు మనం చెప్పాలి. అంటే RAM address లను ప్రోగ్రాంలో ఇవ్వాలి. C లాంగ్వేజ్‌లో ఒక variable కు ముందు ampersand (&) symbol పెడితే, ఆ variable కు కేటాయించిన RAM memory cell number (or) address ను తెలియజేస్తుంది.

ప్రోగ్రాంలో కేటాయించిన memory cell numbers ను మాత్రమే scanf లో గానీ, ఇంకా ఎక్కడకైనా గానీ వాడాలి. అలా కాకుండా వేరే అంకెలను వాడితే Program run అవుతున్నప్పుడు, run-time error (bug) వస్తుంది. ఉదాహరణ: scanf మూడు పూర్ణ సంఖ్యలను రీడ్ చేసి a, b, c అనే మూడు variables లో పెట్టమని చెప్పడానికి వాటి address లను ఇవ్వాలి. ఉత్తరంపై ఎవరి చిరునామా రాస్తే వారికే వెళుతుంది. ఇక్కడ కూడా అదే నియమం వర్తిస్తుంది. scanf statement మూడు integers ను కీబోర్డు నుంచి రీడ్ చేసి, a, b, c అనే variables లో పెట్టడానికి కిందివిధంగా రాస్తాం.

scanf ("%d%d%d" &a, &b, &c);

అదేవిధంగా amount, rate, time అనే variables లోకి మూడు float values ను రీడ్ చేయడానికి

Scanf ("%f%f%f", &amount, &rate , &time); అని రాస్తాం.

3) Calculations చేయడం

రీడ్ చేసిన values తో ఏం కాలిక్యులేట్ చేయాలనేది మన అవసరంపై ఆధారపడి ఉంటుంది. ఏ Operators ఉపయోగించాలి? వాటి అర్థాలేమిటి? తదితర అంశాల గురించి తర్వాతి పాఠ్యాంశాల్లో తెలుసుకుందాం.

4) Results Print చేయడం

ప్రోగ్రాం పూర్తయిన తర్వాత కాలిక్యులేట్ చేసిన ఫలితాన్ని program screen పై Print చేసుకోవచ్చు. Program కరెక్ట్‌గా ఉందా లేదా అనేది Print చేయడం ద్వారానే తెలుస్తుంది. ఉదాహరణకు రేపు పరీక్ష ఉందనుకోండి. ఈ రోజు రాత్రి మనం బాగా చదివాం. అదృష్టం కొద్దీ చదివిన ప్రశ్నలే వచ్చాయి. జవాబులు తెలిసినా సమాధానపత్రంలో రాయలేదు. అప్పుడు మనకు ఎన్ని మార్కులు వస్తాయి?

సున్నా.

ఎందువల్ల?

మనం రాయలేదు కాబట్టి.

అప్పుడు మీరు నాకు జవాబు తెలుసు కదా.. అయినా మార్కులు ఎందుకు

వేయలేదు అనుకుంటే కుదరదు. మీకు సమాధానం తెలుసని మూల్యాంకనం చేసేవారికి మీరు రాస్తేనే కదా తెలిసేది. అదేవిధంగా Program కూడా మనం అడిగితేనే, ఫలితాలను screen పై Print చేస్తుంది. ప్రోగ్రాంకు సంబంధించిన సమాధానాలు RAM లో ఉంటాయి. కానీ మనం అడిగితేనే screen పై Print చేస్తుంది. కాబట్టి తెరపై Print చేయడానికి ఉపయోగించే Printing statements కు ప్రాధాన్యం ఉంది.

ప్రోగ్రాం ఔట్‌పుట్‌లో variable values ను Print చేయాలంటే scanf లా format string ఇవ్వాలి. దీని కోసం variables పేర్లను రాస్తే సరిపోతుంది. వాటి address లను ఇవ్వాల్సిన అవసరం లేదు. ఇది print function requirement (దీన్ని అభివృద్ధి చేసినవారు, అలా రూపొందించారు.)

ఉదాహరణ:

* a, b, c అనే మూడు integer variables విలువలను తెరపై print చేయడానికి కింది printf statement లను ఉపయోగించాలి.

```
printf("%d %d %d\n", a, b, c);
```

* విలువలకు మధ్యలో tab కావాలంటే, ఏ '\t' format string తో కింద చూపినవిధంగా వాడవచ్చు.

```
printf("%d\t %d\t %d\n", a, b, c);
```

* ఇలా కాకుండా, మూడు variable values తెరపై ఒక్కో లైన్‌లో రావాలంటే

```
printf("%d\n %d\n %d\n", a, b, c);
```

* format string లో message లు కూడా రాసుకోవచ్చు.తెరపై "a, b, c values are =" values అని రావాలంటే, printf ను ఇలా రాయాలి.

```
printf("a, b, c values are = %d, %d, %d\n", a, b, c);
```

* అదేవిధంగా a, b ల విలువలను print చేయడానికి

```
printf("a = %d, b = %d, c = %d\n", a, b, c);
```

అని రాయాలి.

మొదటి %d దగ్గర a విలువ, రెండో %d దగ్గర b విలువ print అవుతాయి. ఉదాహరణకు a, b, c విలువలు 10, 20, 30 అనుకుంటే, తెరపై కిందివిధంగా ఫలితం వస్తుంది.

```
a = 10, b = 20, c = 30
```

ఈవిధంగా printf ను వాడుకొని screen పై ఫలితాలను print చేసుకోవచ్చు.

ఇప్పటి వరకూ మనం నేర్చుకున్న అంశాల ఆధారంగా ఒక ప్రోగ్రాం రాసే విధానాన్ని పరిశీలిద్దాం.

a) ఎన్ని values read చేయాలి? ఏం కాలిక్యులేట్ చేయాలి? అనే అంశాల ఆధారంగా ముందుగా variables ను డిక్లేర్ చేయాలి.

b) scanf ను ఉపయోగించి డేటా read చేయడం.

c) ఇచ్చిన ప్రోగ్రాం కోడ్ ఆధారంగా రీడ్ చేసి వాటితో కాలిక్యులేషన్స్ చేయాలి.

d) కాలిక్యులేట్ చేసినవాటిని screen పై రాయమని printf ద్వారా చెప్పాలి.

పైన చెప్పిన స్టేట్‌మెంట్లతోపాటు ప్రోగ్రామర్‌కు కొన్ని సూచనలు చేయడానికి instructions ను వాడతాం. వీటిని user interface statements అంటారు. ఇవి లేకపోయినా program పని చేస్తుంది. కానీ ఒకరు రాసిన ప్రోగ్రాం మరొకరు పరిశీలించినా అర్థమవడానికి ఇవి ఉపయోగపడతాయి.

ఉదాహరణ: మూడు పూర్ణసంఖ్యలను (Integer Values) కీబోర్డు నుంచి enter చేయమని చెప్పడానికి scanf statement ముందు కింది స్టేట్‌మెంట్‌ని రాస్తాం.

printf("Enter three integer values\n");

ఇలాంటివాటినే user interface statements అంటారు.

ఇప్పుడు మనం ఒక చిన్న ప్రోగ్రాం రాద్దాం. ముగ్గురు విద్యార్థుల మార్కులను కీబోర్డు నుంచి read చేసి వాళ్ల మార్కుల సరాసరిని తెరపై print చేయాలనుకుందాం.

i) దీని కోసం మూడు int type variables తోపాటు ముగ్గురి మార్కులను నిక్షిప్తం చేయడానికి ఒక float variable కావాలి. ఎందుకంటే సరాసరి ఎప్పుడూ పూర్ణ సంఖ్యగా ఉంటుందని చెప్పలేం. మొత్తం మీద నాలుగుvariables అవసరం.

ii) Enter three students marks అని User Interface statement రాయాలి.

iii) scanf కు ముందు integers ను కీబోర్డు నుంచి రీడ్ చేసి variables లో నిక్షిప్తం చేయమని చెప్పాలి. ఒకవేళ నాలుగు variables కు value చేస్తే ఎలా ఉంటుంది? ప్రశ్నతోపాటు సమాధానం ఇచ్చినట్లు ఉంటుంది. సరాసరిని మనం కాలిక్యులేట్ చేయాలి. ఏ మూడు మార్కులు ఇచ్చినా, మనకు వాటి సరాసరి కావాలి.

iv) మనం ఇచ్చిన మూడు విలువల సరాసరిని కాలిక్యులేట్ చేయమని చెప్పే statement రాయాలి.

v) calculate చేసిన సరాసరిని screen పై ప్రింట్ చేయమని చెప్పాలి.

ఈ సూచనలను పాటిస్తూ రాసిన ప్రోగాం, దాని output కింద చూపిన విధంగా ఉంటుంది..

Program run అవుతున్నప్పుడు విలువల మధ్యలో space, TAB, new line లలో ఏదో ఒకటి ఇవ్వాలి. ఇక్కడ ఇచ్చిన Program లో మూడు printf statement లను చివరలో ఉపయోగించాం. మనకు screen పై ఏవిధమైన ఫలితం కావాలో, ఆవిధంగా printf ను ఉపయోగించవచ్చు. అంతేగానీ మూడూ ఒకేసారి ఉపయోగించాల్సిన అవసరం లేదు.

```
≡ File Edit Search Run Compile Debug Project Options Window Help
\NBVAVG.C
#include<stdio.h>
int main(){
 int a,b,c;   /* Variable declarations. */
 float avg;

 /* User interface statment*/
 printf("[illegible]");

 /* Data reading statement */
 scanf("[illegible]",&a,&b,&c);

 /* Calculation of average */
 avg = (a+b+c)/3;

 /* Statements to print results*/
 printf("[illegible]", avg);
 printf("[illegible]",a,b,c,avg);
 printf("[illegible]", avg,a,b,c);

 return 0;
}
19:1
F1 Help  Alt-F8 Next Msg  Alt-F7 Prev Msg  Alt-F9 Compile  F9 Make  F10 Menu
```

Program development under Turbo C

```
Enter three students marks
10 30  90
43.000000
Average of 10  30  90 = 43.000000
43.000000 is the average of 10, 30 90
_
```

Sample results Screen

'సీ' ప్రోగ్రామ్‌తో సున్నం వేయండి !

* * 'సీ' లాంగ్వేజ్‌లో వివిధ రకాల వేరియబుల్స్ ని ఉపయోగించి ప్రోగ్రాం రాయడం



ఇంతకు ముందు పాఠంలో సీ ప్రోగ్రాం రాయడం నేర్చుకున్నాం. అందులో integer టైపు వేరియబుల్స్ ను ఎలా వాడాలో తెలుసుకున్నాం. ఈ పాఠంలో వివిధ రకాల వేరియబుల్స్ ని ఉపయోగించి ప్రోగ్రాం ఎలా రాయాలో నేర్చుకుందాం. ఉదాహరణకు Students మార్కులను Process చేయడానికి integer Type variables ఉపయోగిస్తాం.

* సాధారణంగా స్టూడెంట్ మార్కులు integer (పూర్ణ సంఖ్యలు) రూపంలో ఉంటాయి. ఏ variable ను ఎప్పుడు వాడాలో అనేది ఆ variables లో మనం ఎంత పెద్ద నెంబరు ఇవ్వాలనుకుంటున్నామో అనే దాని మీద ఆధారపడి ఉంటుంది. Character టైపు వేరియబుల్స్ ను ఎప్పుడు వాడాలో తర్వాతి పాఠంలో తెలుసుకుందాం.

ఈ కింద ఇచ్చిన Table, సీ లాంగ్వేజిలో వున్న వేరియబుల్స్ వివరాలు.

Variable type	SIZE (Bits)	Range
char or signed char	8	-128 to 127
unsigned char	8	0 to 255
int or signed int	16	-32768 to 32767
unsigned int	16	0 to 65535
short int or signed short int	8	-128 to 127
unsigned short int	8	0 to 255
long int or signed long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
float	32	3.4 e-38 to 3.4 e+38
double	64	1.7e-308 to 1.7e+308
long double	80	3.4 e-4932 to 3.4 e+4932

- scanf, printf లలో ఏ formats లో ఏ Variable ఉపయోగించాలో వివరించే పట్టిక

%c – Print an character
%d – Print an Integer
%i – to Print/read an Integer
%e – to Print/read float value in exponential form.
%f – to Print/read float value
%o – to Print/read octal value
%s – to Print/read a string
%x – to Print/read a hexadecimal integer (unsigned) using lower case a – F0
%X – to Print/read a hexadecimal integer (unsigned) using upper case A – F
%a – to Print/read a unsigned integer.
%p – Print a pointer value
%hx – to Print/read hex short
%lo – to Print/read octal long

%ld – to Print/read long

ఉదాహరణ:

ఒక గది పొడవు, వెడల్పు, ఎత్తులను తీసుకొని, గది పరిమాణాన్ని, సున్నం వేయాల్సిన వైశాల్యాన్ని ప్రింట్ చేసే విధంగా ప్రోగ్రాం రాయండి.

Note: గది నేలభాగానికి ఎవరూ సున్నంవేయరు. కాబట్టి దాన్ని లెక్కలోకి తీసుకోకూడదు.

ప్రోగ్రాం రాసే విధానం:

* ముందుగా మనం డిజైన్ చేయాల్సిన Variables ఎన్నో రాసుకోవాలి
* మొత్తం ఐదు Variables ను అవసరం అవి length, breadth, height volume, area
* సాధారణంగా గది పరిమాణాలు fractionals లో ఉండే అవకాశం ఉంది. కాబట్టి float టైప్ Variable ఉపయోగించాలి.
* గది length, breadth, height లను రీడ్ చేసి volume (పరిమాణం)ను, area (వైశాల్యం)ను ప్రింట్ చేసే విధంగా ప్రోగ్రాం రాయాలి.

Program:

```
#include<stdio.h>
int main()
{
float l, b, h, v, sa;
printf("Enter length, breadth, height\n");
scanf("%f%f%f", &l, &b, &h);
v = l*b*h;
sa = 2*(l+b)*h+(l*b); // surface area
printf("Volume=%f\nSurface area to be white washed=%f\n ",v, sa);
return (0);
}
```

- Output :

```
C:\area.exe
Enter length, breadth, height
12.33 10.34  8.92
Volume = 1137.230469
Surface area to be white washed = 531.924988
```

- మనం విలువలను కీబోర్డ్ నుంచి ఇచ్చేటప్పుడు విలువకి, విలువకి మధ్య టాబ్ స్పేస్ ఇవ్వచ్చు. అంతే కాకుండా ఒక్కో విలువ ఒక్కో లైనులో కూడా ఇచ్చే అవకాశం ఉంది. %f format ని scanf లో వాడినా కూడా Input ఇచ్చేటప్పుడు integer విలువలను ఇచ్చినా తీసుకుంటుంది. ఎందుకంటే పూర్ణ సంఖ్యలు, సహజ సంఖ్యలోని భాగాలే కాబట్టి.మనం రాసిన ప్రోగ్రాం లో, User కు గది length, breadth, height వాల్యులను printf ద్వారా ఇస్తాం. అలా ఇస్తేనే volume ను, area ను కాలిక్యులేట్ చేసే విధంగా ప్రోగ్రాం రాశాం. ఈ ప్రోగ్రాంలో User ఇన్ పుట్ ఇవ్వడం కుదరదు .

User ఇన్‌పుట్ ఇచ్చే విధంగా ప్రోగ్రాం ఎలా రాయలో తెలుసుకుందాం.

Program:

```
#include<stdio.h>
int main()
{
float l, b, h, v, sa;

printf("Enter length:\n");
scanf("%f", &l);

printf("Enter width:\n");
scanf("%f", &b);

printf("Enter height:\n");
scanf("%f", &h);

v = l*b*h;
sa = 2*(l+b)*h+(l*b); // surface area
printf("VVolume=%f\nSurface area to be white
```

```
washed=%f\n",v, sa);
        return(0);
        }
```

- Output :

```
C:\area.exe
Enter length
12.33
Enter breadth
10.34
Enter  height
8.92
Volume = 1137.230469
Surface area to be white washed = 531.924988
```

- ఉదాహరణ:

Fahrenheit temperature ను రీడ్ చేసి దానికి సమానమయిన centigrade temperature ను లెక్కించి చేసి ప్రింట్ చేసే ప్రోగ్రాం రాయండి.

ప్రోగ్రాం రాసే విధానం:

* Temperature విలువలు సాధారణంగా సహజ సంఖ్యలుగా ఉంటాయి కాబట్టి, ప్రోగ్రామర్ float టైప్ వేరియబుల్ ని తీసుకోవాలి.
* రెండు వేరియబుల్స్ అవసరం ఒక వేరియబుల్ రీడ్ చేసిన దానిని స్టోర్ చేయడానికి,మరో వేరియబుల్ లెక్కించిన దాన్ని స్టోర్ చేయడానికి.
* Fahrenheit నుంచి centigrade కు మార్చే విధంగా ప్రోగ్రాం రాయాలి. దీని కోసం మనం physics లో ఉన్న Fahrenheit నుంచి centigrade కు మార్చే ఫార్ములా ఉపయోగించాలి

```
#include<stdio.h>
int main()
{
float it, ot;
printf("Enter Fahrenheit temperature value\n");
scanf("%f", &it);
ot = 5/9*(it-32);
printf("%f\n", ot);
return(0);
}
```

- Output:

```
C:\area.exe
Enter Fahrenheit temperature value
98.04
0.000000
```

- Note:

మనం 98.04 ఇన్‌పుట్ ఇస్తే సున్నా రిజల్ట్స్ వచ్చింది. మనకు తెలిసి 36.6 రావాలి. రావడములేదు ఎందుకు?

```
C:\area.exe
Enter Fahrenheit temperature value
36.06
0.000000
```

* ఇంకో వాల్యూ 36.06 ఇచ్చి రన్ చేస్తే కూడా సున్నా వచ్చింది?
* ఎందుకు? ప్రోగ్రాములో ఎక్కడా మిస్టేక్ లేదు? మరి ఎందుకిలా వస్తుంది

Integer Mode and Float Mode Arithmetic's

అన్ని ప్రోగ్రామింగ్ లాంగ్వేజీల్లో కూడా, ఏవైనా operator ను ఎవాల్యుయేట్ చేస్తున్నప్పుడు, అన్ని operands integer టైపు అయితే రిజల్ట్ కూడా integer అవుతుంది. అంటే 10/3 అనేది 3 అవుతుంది. అలాగే 1/3 లేదా 5/9 లు 0 (సున్నా)లు అవుతాయి. ఈ కారణం వల్ల , పై ప్రోగ్రాంలో ఏ విలువ ఇచ్చినా సున్నా వస్తోంది. దీన్ని సరి చేయాలంటే operands లో ఒక్క operand అయినా float (లేక double) అవ్వాలి. అంటే 5/9 ఏమో సున్నా అవుతుంది, అదే 5/9.0 సున్నా అవ్వదు. ఇది ప్రతీ ప్రోగ్రామింగ్ లాంగ్వేజి లో ఉండే స్వభావం.

Type casting

- ఒక Variable, Constant, Expression లను other టైప్ లోకి మార్చాలంటే వాటిని ముందు బ్రాకెట్ లో ఏ టైపు లోనికి మార్చాలని అనుకుంటున్నామో రాయాలి. దీనినే type casting అంటారు.

ఉదాహరణకు

(float) 5 అనేది 5.0 అవుతుంది.
(int) 5.7 అనేది 5 అవుతుంది. దీనిని వాడి పై ప్రోగ్రాంని సరి చేయవచ్చు
పై ప్రోగ్రాంలో physics లో ఉన్న equation న్నిఉన్నది ఉన్నట్టుగా మార్చినందు వల్ల సమస్య వచ్చింది. పైన ఇచ్చిన దానిలో ot=5*(it-32)/9; అని రాసి ఉంటే సమస్య వచ్చేది కాదు. ఎందుకంటే బ్రాకెట్ లోది float అవుతుంది. దాన్ని 5 తో హెంచితే కూడా float అవుతుంది. ఆ వచ్చిన దానిని 9 తో డివైడ్ చేస్తే కూడా float

అవుతుంది. ఫైనల్ గా float వాల్యూ వస్తుంది.
పైన ఇచ్చిన ప్రోగ్రాములో, ot = 5/9*(it-32); అనే దాని బదులుగా ఈ కింద ఇచ్చిన వాటిని వాడి సరైన ఫలితాన్ని సాధించవచ్చు.

ot = 5*(it-32)/9;	First (it-32) is calculated. As it is float, it-32 becomes float. Then, 5*(it-32) gives float. When we divide with 9, we still get float.
ot = 5.0/9*(it-32);	5.0/9 results is float. Thus, final result is float.
ot = 5/9.0*(it-32);	5/9.0 results is float. Thus, final result is float.
ot = 5.0/9.0*(it-32);	5.0/9.0 results is float. Thus, final result is float.
ot = ((float)5/9)*(it-32);	(float) 5 will be taken as 5.0. Thus, 5.0/9 results is float. Thus, final result is float.
ot = (5/(float)9)*(it-32);	(float) 9 will be taken as 9.0. Thus, 5/9.0 results is float. Thus, final result is float.

-

Output:

```
C:\area.exe
Enter Fahrenheit temperature value
36.06
2.255556
```

ఉదాహరణ

విద్యార్థి మార్కులను తీసుకొని average ప్రింట్ చేయాలి. ఫలితం integer లో కాకుండా float లో వచ్చే విధంగా ప్రోగ్రాం రాయండి

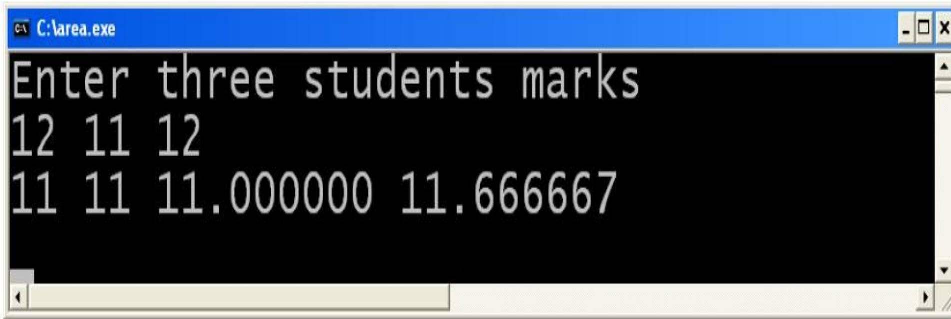
- **ప్రోగ్రాం రాసే విధానం:**

* ఫలితం float లో రావాలంటే, LHS(left hand side) variable float టైప్ అవ్వాలి, RHS(right hand side) expression కూడా float mode లో ఎవాల్యుయేట్ అవ్వాలి.
* ప్రోగ్రాంలో నాలుగు రిజల్ట్ వేరియబుల్స్ avg1,avg2, avg3, avg4 లను ఉపయోగించాలి.
* మూడు వేరియబుల్స్ (a, b, c) ను ఇన్‌పుట్ ఇచ్చిన వాటిని స్టోర్ చేయడానికి, ఇంకో Variable (s) వాటి మొత్తాన్ని స్టోర్ చేయడానికి ఉపయోగించాలి.
* avg1=s/3; అనే దాన్లో రెండూ integer టైప్ కాబట్టి ఫలితం integer లో వస్తుంది. అలాగే, avg2=s/3.0; RHS float లో ఎవాల్యుయేట్ చేసినా avg2 integer కాబట్టి అందులో integer వాల్యూ సేవ్ అవుతుంది. అలాగే, avg3=s/3; అనే దానిలో రెండూ integer టైప్ కాబట్టి ఫలితం integer లో వస్తుంది. avg4=s/3.0; RHS float లో ఎవాల్యుయేట్ చేసి వచ్చిన float

దానిని avg4 లో స్టోర్ అవుతుంది.

Program

- ```
  #include<stdio.h>
  int main()
  {
  float avg3, avg4;
  int a, b, c, s, avg1, avg2;
  printf("Enter three students marks");
  scanf("%d%d%d", &a, &b, &c);
  s = a+b+c;
  avg1=s/3;
  avg2=s/3.0;
  avg3=s/3;
  avg4=s/3.0;
  printf("%d %d %f %f\n", avg1, avg2, avg3, avg4);
  return (0);
  }
  ```
- Output :



- ఉదాహరణ

ఒక వ్యక్తి నెల Basic Salary ,DA,TA, CCA, HRA , బోనస్ లను రీడ్ చేసి, అతని నెలవారీ మొత్తం Salary ని ప్రింట్ అయ్యే విధంగా ప్రోగ్రాం రాయండి.

ప్రోగ్రాం రాసే విధానం

* సాధారణంగా, DA,TA, CCA, HRA లను, Basic Salary మీద శాతంగా పరిగణిస్తారు. ఇక్కడ వీటన్నింటిని integers గా తీసుకోవాలి.

ఉదాహరణకు

ఈ కింది విలువలు తీసుకుంటే,

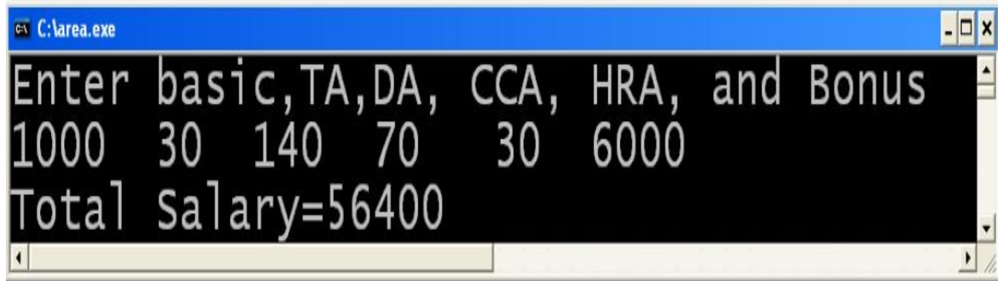
Basic TA DA CCA HRA BONUS
1000 60 110 35 75 6000
  ```

- TA 600 (1000*60/100) అవుతు0 ది, DA 1100 అవుతు0 ది, CCA 350 అవుతుంది, మరియూ HRA 750 అవుతుంది. అలాగే, total monthly salary = 1000+600+1100+350+750=3800 అవుతుంది. ta,da,cca, hra లను integer లాగా తీసుకొన్నాం కాబట్టి 100.0 ను total calculation లో వాడాలి. లేకపోతే, ఫలితం సరిగారాదు.

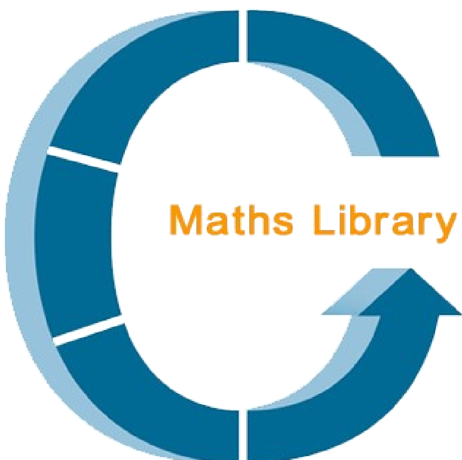
Programe:

```
#include<stdio.h>
int main()
{
int basic, da, ta, cca, hra, bon, tot ;
printf("Enter basic,TA,DA, CCA, HRA, and Bonus\n");
scanf("%d%d%d%d%d%d", &basic, &ta, &da, &cca, &hra, &bon);
tot =12*basic*(1+(ta+da+cca+hra)/100.0)+2*bon;
printf("Total Salary=%d\n", tot);
return(0);
}
```

Output :



- గణిత గ్రంథాలయాన్ని ఎలా వాడాలి ?



- సీ ప్రోగ్రాంలో మ్యాథ్స్ లైబ్రరీ చాలా ముఖ్యమైంది. మనం ప్రోగ్రాంలో sine, cos, tan, etc.. లాంటి మ్యాథమెటిక్ సూత్రాలను ఉపయోగించడానికి మ్యాథ్స్లైబ్రరీ అవసరం. ఈ మ్యాథమెటిక్ విలువలను ఉపయోగింపచడానికి అవసరమైన ప్రీడిఫైన్ (ముందుగా రాసి ఉంచిన) ప్రోగ్రాంలు ఈ లైబ్రరీలో రాసి ఉంటాయి. కేవలం Operator ని మాత్రమే మ్యాథ్స్ లైబ్రరీ అవసరం లేకుండా ఉపయోగించగలం. మాథ్స్ లైబ్రరీలోని ఫంక్షనులను వాడుకొని కొన్ని చిన్న ప్రోగ్రాములను ఎలా

రాయలో ఈ పాఠంలో తెలుసుకుందాం. మాథ్స్ కి సంబంధించిన రెడీమేడ్ ప్రోగ్రామ్‌లు (ఫంక్షనులు) చాలా వున్నాయి. వాటిని ఉపయోగించకుండా మనం చాలా ప్రోగ్రాంలు రాయడం కష్టం. ఈ ఫంక్షన్స్‌ని వాడాలంటే, ముందుగా #include<math.h> అనే లైను మన ప్రోగ్రాంలో రాయాలి.

ఈ మాథ్స్ లైబ్రరీలో చాలా ఫంక్షనులు వున్నాయి. ఉదాహరణకు, sqrt, log10, pow, sin, cos, etc. . వీటిని ఎలా వాడి ప్రోగ్రాములు రాయాలో నేర్చుకుందాం. మాథ్స్ లో వాడే xy ని, సీ లాంగ్వేజ్ లో రాయలంటే pow అనే ఫంక్షన్ ఉపయోగించి pow(x,y) అని రాస్తాం.

Note: ఈ ఫంక్షన్‌లను ప్రోగ్రాంలో ఉపయోగిస్తే మరచి పోకుండా #include<math.h> అనే లైను మన ప్రోగ్రాంలో రాయాలి.

ఉదాహరణ: ప్రిన్సిపల్ మొత్తం (p), ధర (r), సమయం (t) సంవత్సరాలలో తీసుకొని సాధారణ వడ్డీ (simple interest), చక్ర వడ్డీ (compound interest) లెక్కించి ప్రింట్ చేసేలా ప్రోగ్రామ్ రాయండి ?

సమాధానం:

* అవసరమైన ఫార్ముల్లాలు రాసుకోండి

Simple interest=prt / 100

Compound interest= $p*(1+r/100)^t - p$

- ప్రోగ్రాం:
-
```
#include<stdio.h>
#include<math.h>
                    int main()
                    {

                    float p,r,t,simple,compound;
                    printf("Enter principal amount, rate and
time duration\n");
                    scanf("%f%f%f",&p,&r,&t);
                    simple=p*r*t/100;
                    compound=p*pow(1+r/100.0,t)-p;
        printf("Simple Interest=%f\nCompound
Interest=%f\n", simple, compound);
                    return (0);
}
```

- Output:

```
C:\area.exe
Enter principal amount, rate and time duration
1000 12.5 2.5
Simple Interest=312.500000
Compound Interest=342.398041
```

ఉదాహరణ

- ఒక కంపెనీ నెలసరి డిపాజిట్ స్కీమ్ లాంటి పథకాల్లో ఎంత డిపాజిట్ చేస్తే మెచ్యూరిటీ విలువ ఎంత అవుతుందో తెలుసుకోవడానికి ప్రోగ్రాం రాయండి ?
- ప్రోగ్రాం రాసే విధానం
- * ఈ ప్రోగ్రాంకి నెలసరి వాయిదా వాల్యూ (R), సంత్సరం వడ్డీ రేటు(rr), ఎన్ని క్వార్టర్ (n n) లు ఇన్ స్టాల్ మెంట్ పే చేస్తామో Input లాగా యివ్వాలి.
  * ఉపయోగించాల్సిన ఫార్ములా
  Maturity = (R * [(1+r)^{n} - 1]) / (1-(1+r)$^{-1/3}$)
- **ప్రోగ్రాం:**

```
#include<stdio.h>
#include<math.h>
                    int main()
                    {
                    float R,r,n,Maturity;
                    printf("Enter Monthly Installment, rate
and number of quarters\n");
                    scanf("%f%f%f",&R,&r,&n);

                    Maturity=(R * (pow(1+r/400, n) - 1)) /
(1-pow(1+r/400,-1/3.0) );
                    printf("Maturity Amount=%f\n",
Maturity);
return (0);
}
```

- **Output:**

```
C:\area.exe
Enter Monthly Installment, rate and number of quarters
1000 10 4
Maturity Amount=12664.602539
```

ఉదాహరణ: ఒక త్రిభుజం మూడు మూడు భుజాల కొలతలు (a, b, c) ఇన్పుట్గా ఇచ్చి త్రిభుజ

వైశాల్యాన్ని లెక్కించే విధంగా ప్రోగ్రాం రాయండి

ప్రోగ్రాం రాసే విధానం

* ప్రోగ్రాం రాయడానికి మనకు అవసరమైన సూత్రాన్ని రాసుకోవాలి.
* అవసరమైన సూత్రం

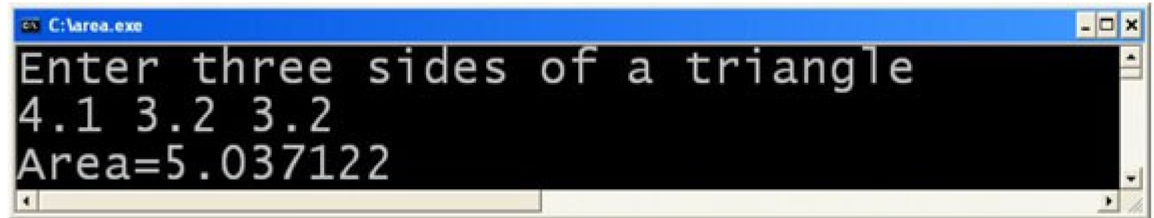
$$S = (a+b+c)/2$$

$$\text{Area} = \sqrt{S(S-a)(S-b)(S-c)}$$

- ప్రోగ్రాం:

```
#include<stdio.h>
#include<math.h>
        int main()
        {
        float a,b,c,s,area;
        printf("Enter three sides of a triangle\n");
        scanf("%f%f%f",&a,&b,&c);
        s=(a+b+c)/2.0;
        area=sqrt(s*(s-a)*(s-b)*(s-c));
        printf("Area=%f\n", area);
        return (0);
}
```

- Output:



- ఉదాహరణ: ఒక integer ను ఇన్‌పుట్‌లా తీసుకొని, దానిలో ఎన్ని అంకెలున్నాయి divide చేయగలగిన అతి పెద్ద 10^p లాగా ఉండే సంఖ్యను ప్రింట్ అయ్యేలే ప్రోగ్రాం రాయండి. ?

ఈ ప్రోగ్రాంలో p అనేది ఓ integer.

ప్రోగ్రాం రాసే విధానం

* ఈ ప్రోగ్రాంలో p అనేది ఒక integer. ఉదాహరణకు 178 ఇన్‌పుట్ లా ఇస్తే, ఫలితం 3, 100. 1238 ఇన్‌పుట్ లా ఇస్తే ఫలితం 4, 1000.
12538 ఇన్‌పుట్ ఇస్తే ఫలితం 5, 10000. వచ్చే విధంగా ప్రోగ్రాం రాయాలి.
* ప్రోగ్రాం ఫలితం రావాలంటే, ఈ కింద ఇచ్చిన వాటిని గమనించండి

log10(10)=1
log10(99)=1.9999999999

- log10(100)=2
  log10(999)=2.9999999999
- log10(1000)=3
  log10(9999)=3.9999999999
- log10(10000)=4
  log10(99999)=4.9999999999
- log10(100000)=5
  log10(999999)=5.9999999999

పైన ఇచ్చిన వాటి నుంచి గమనిస్తే . ఏ సంఖ్యకు log10 అప్లై చేస్తే వచ్చిన విలువ integer భాగము విలువకు ఒకటి కలిపితే ఆ సంఖ్యలో ఉన్న అంకెలెన్నో తెలుస్తాయి.

పైన ఇచ్చిన వాటి నుంచి గమనిస్తే . ఏ సంఖ్యకు log10 అప్లై చేస్తే వచ్చిన విలువ integer భాగము విలువకు ఒకటి కలిపితే ఆ సంఖ్యలో ఉన్న అంకెలెన్నో తెలుస్తాయి.

* అలాగే రెండో దానికి కూడా ఇక్కడే సమాధానం ఉంది. ఇన్‌పుట్ 178 ఇస్తే మనకు అవుట్‌పుట్ 100 రావాలి. అంటే 10^2 రావాలి. 178 log10 integer భాగం 2.

* 1238 ఇస్తే మనకు 1000 రావాలి అంటే 10^3 రావాలి. 1238 log10 integer భాగం3.

* ఈ ప్రోగ్రాంలో మనం ఇచ్చిన సంఖ్య log10 integer భాగం (p) కనుక్కొని, దానికి 10^p విలువను లెక్కించి Output వచ్చేలా ప్రోగ్రాం రాయాలి.

- ప్రోగ్రాం:

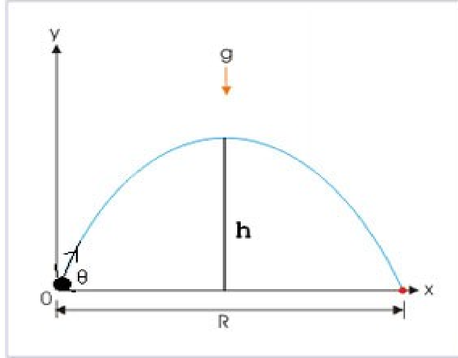
```
#include<stdio.h>
#include<math.h>
                int main()
                {
                int n,p,nd,F;
                printf("Enter an integer\n");
                scanf("%d",&n);
                p=log10(n);
                nd=p+1;
                F=pow(10,p);

                printf("Number of Digits=%d\nLargest
Divisor=%d\n", nd,F);
                return (0);
}
```

- Output:

```
C:\area.exe
Enter an integer
1238
Number of Digits=4
Largest Divisor=1000
```

ఉదాహరణ: ఒక బంతిని θ కోణంతో పైకి విసిరితే ఎంత ఎత్తుకు వెళుతుంది. అది ఎంత దూరం పడుతుందో లెక్కించే విధంగా ప్రోగ్రాం రాయండి ?



ప్రోగ్రాం రాసే విధానం:

* ప్రోగ్రాంకు అవసరమైన సూత్రాలు

$$R = \frac{u^2 * \sin 2\theta}{g}$$

$$h = \frac{u^2 * \sin^2\theta}{2 * g}$$

- * కోణాన్ని Degrees, minutes, seconds లలో ఇస్తాం కాని Output లో దూరం తెలుసుకోవడానికి ఉపయోగించే సూత్రంలో radians విలువ ఇవ్వాలి. కాబట్టి Degrees ని radians లోకి మార్చాలి. తర్వాత h, R కాలిక్యులేట్ చేయాలి.

  Note: g విలువను 9.8 గా తీసుకోవాలి .

  3.14 radians = 180 Degrees

  1 Degree = 60minutes

  1 Minute = 60 seconds

- **ప్రోగ్రాం:**

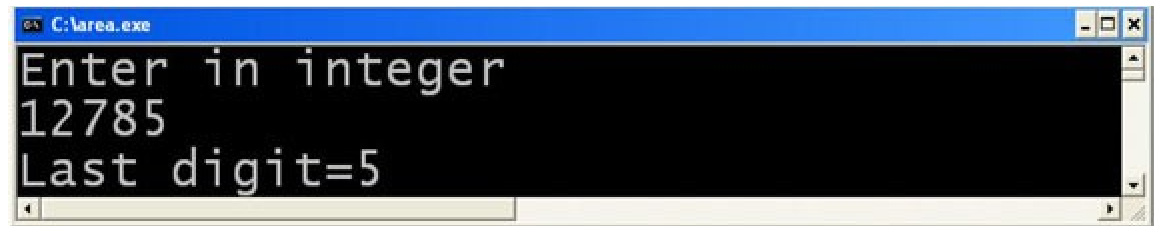
```
#include<stdio.h>
#include<math.h>
        int main()
        {
        int d,m,s;
        float u, h,R,a;
```

```
            printf("Enter angle in degrees,minutes
and second\n");
            scanf("%d%d%d",&d,&m,&s);
            printf("Enter initial velocity\n");
            scanf("%f",&u);
            a=(d+m/60.0+s/3600.0)*3.14/180.0;

            h=u*u*sin(a)*sin(a)/(2*9.8);
            R=u*u*sin(2*a)/9.8;

            printf("Maximum Height=
%f\nMaximum Horizontal Distance=%f\n", h, R);
            return (0);
}
```

- Output:
-



- మాథ్స్ లైబ్రరీలో cos, tan,sin, etc., కూడా ఉంటాయి. మనం గుర్తుంచుకోవాల్సిన విషయం వీటన్నింటికి కూడా కోణాన్ని radians లో ఇవ్వాలి
  abs, fabs, dabs అనేవి కచ్చితమైన విలువను కనుక్కోవడానికి ఉపయోగిస్తాం.
  * ఒక integer కచ్చితమైన విలువను కనుక్కోవడానికి abs వాడతాం.
  * float కచ్చితమైన విలువను కనుక్కోవడానికి fabs ఉపయోగిస్తాం
  * double కచ్చితమైన విలువను కనుక్కోవడానికి dabs వాడతాం.
  ఉదాహరణకు, int x=7, y=-19; అయితే abs(x) విలువ 7 అవుతుంది. abs(y) అనేది 19 అవుతుంది.
  float x=7.12, y=-19.78; అయితే abs(x) విలువ 7.12 అవుతుంది. abs(y) విలువ 19.78 అవుతుంది.
- Modulus Operator:
  ఇప్పటి వరకు, మనం +,-,*,/ లను ఎలావాడాలో నేర్చుకొన్నాం. అలాగే, సీ లాంగ్వేజిలో మరో operator ఉంది. అదే modulus operator(%). ఇది కూడా binary operator, అంటే దీనికి కూడా రెండు operands ఉంటాయి. రెండూ operands కూడా integer టైపుగా ఉండాలి. A, B లు రెండు integer operands అయితే, A %B=|A| ను |B| divide చేస్తే మిగిలేది. ఫలితానికి సంబంధించి sign A లో sign

లాగా ఉంటుంది. ఉదాహరణకు 10%3=1, 10%-3=1, -10%3=-1, -10%-3=-1., A%B కాబట్టి సున్న అయితే B ను A కి factor అని అంటాం. ఈ operator ను వాడాలి అంటే <math.h> ను వాడాల్సిన అవసరము లేదు. ఇది ఫంక్షను కాదు. ఇది operator.

ఒక integer ను ఇన్‌పుట్ లా తీసుకొని దాని చివరి అంకెను లెక్కించేవిధంగా ప్రోగ్రాం రాయండి

ఉదాహరణ:

ఒక integer ను ఇన్‌పుట్ లాగా తీసుకొని దాని చివరి అంకెను లెక్కించేవిధంగా ప్రోగ్రాం రాయండి ?

ప్రోగ్రాం రాసే విధానం:

* ఈ ప్రోగ్రాం రాయడానికి ఇచ్చిన నెంబర్‌కి 10 తో modulus అప్లై చేస్తాం

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
{
int dig,n;
printf("Enter in integer\n");
scanf("%d",&n);
dig=n%10;
printf("Last digit=%d\n", dig);
return (0);
}
```

డెసిషన్ మే'కింగ్' రిలేషనల్ ఆపరేటర్లు !



'సీ' లాంగ్వేజి లో వున్న రిలేషనల్ ఆపరేటర్లు బైనరీ ఆపరేటర్లు, అవి రెండు operands ను కలిగి ఉంటాయి. ఆ రెండు operands, వేరియబుల్స్ అయినా అవ్వవచ్చు, లేదా constants అయినా కావచ్చు లేదా expressions అయినా కావచ్చు . ఈ రిలేషనల్ ఆపరేటర్ల ఫలితం విలువ సత్యం (True) లేదా అసత్యం (False) అవుతుంది. దానినే సంఖ్యాపరంగా 1 లేదా 0 (సున్నా) అని చెప్పవచ్చు. అంటే సత్యం అయితే 1, అసత్యం అయితే 0 అవుతుంది.

గమనిక: సీ లాంగ్వేజిలో ఏదైనా పాజిటివ్ లేదా నెగెటివ్ విలువను logical గా సత్యం అనుకుంటుంది, సున్నాను అసత్యం అనుకుంటుంది.

'సీ 'లాంగ్వేజి లో ఈ కింద ఇచ్చిన పట్టికలో రిలేషనల్ ఆపరేటర్లు, వాటి ఉదాహరణలు ఉన్నాయి. ఇందులో A,B విలువలను 10,

రిలేషనల్ ఆపరేటరు	Logical వాల్యూ	సంఖ్యాపరమైన విలువ	
A>B	ఫాల్స్	0	ఇక్కడ రెండు వేరియబుల్స్ వాడాము.
A>=B	ఫాల్స్	0	ఇక్కడ రెండు వేరియబుల్స్ వాడాము.
A<10	ఫాల్స్	0	ఇక్కడ ఓ వేరియబుల్, ఓ కానిస్టంట్ వాడాము.
A<=B	ట్రూ	1	ఇక్కడ రెండు వేరియబుల్స్ వాడాము.
A==B	ఫాల్స్	0	ఇక్కడ రెండు వేరియబుల్స్ వాడాము.
A!=10	ఫాల్స్	0	ఇక్కడ ఓ వేరియబుల్, ఓ కానిస్టంట్ వాడాము.
((A+B)==0)	ఫాల్స్	0	ఇక్కడ ఓ expression, ఓ కానిస్టంట్ వాడాము.

ఉదాహరణ: 1

మూడు అంకెలు ఉండే ఒక integer ను ఇన్‌పుట్ లాగా తీసుకొని, అది ఇరువైపుల ఒకేవిధంగా వచ్చేపదం (Palindrome) అయితే 1 లేకపోతే 0 అని ప్రింట్ చేసేలా ప్రోగ్రాం రాయండి ?

ప్రోగ్రాం రాసే విధానం

* ఒక సంఖ్యను తిప్పి రాసినా అదే సంఖ్య వచ్చే వాటిని palindrome అని అంటాం. ఉదా: 292, 393, 22322.

* ఇచ్చిన ప్రశ్న ప్రకారం ప్రోగ్రాంలో మూడు అంకెలు ఉండే సంఖ్యలపై మాత్రమే రాయాలి

* మనం ఇచ్చిన నెంబరులోని మొదటి అంకెను 100 తో భాగించి (divide) తర్వాత వచ్చే సంఖ్యను చివరి అంకెతో పోల్చి చూడాలి.

* చివరి అంకె రావాలంటే మనం ఇచ్చిన నెంబర్‌ని 10 తో modulus చేయాలి . రెండు ఒకటే అయితే 1 లేకపోతే 0 (సున్నా) వస్తుంది.

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
    {
    int firstdigit, lastdigit, dec, n;
    printf("Enter an integer\n");
    scanf("%d",&n);
    lastdigit=n%10;
    firstdigit=n/100;
    dec=(firstdigit==lastdigit);
    printf("Result=%d\n", dec);
    return (0);
    }
```

Output:

```
C:\lesson6.exe
Enter an integer
387
Result=0
```

* ప్రోగ్రాంలో 387 ఇన్‌పుట్ గా ఇచ్చాం అది palindrome కాదు కాబట్టి 0 వచ్చింది మరోసారి ప్రోగ్రాం రన్ చేసి ఇన్‌పుట్‌గా 757 ఇవ్వండి. 757 palindrome కాబట్టి 1 వచ్చింది.

```
C:\lesson6.exe
Enter an integer
787
Result=1
```

ఉదాహరణ: 2

విద్యార్థి పరీక్షలో ఒక సబ్జెక్ట్‌లో సాధించిన మార్కులను ఇన్‌పుట్‌గా ఇస్తే పాస్ అయితే మార్కుల ప్రింట్ రావాలి, లేదా 0 (సున్నా) ప్రింట్ చేసే విధంగా ప్రోగ్రాం రాయండి.
గమనిక: ఇక్కడ పాస్ మార్క్స్ 35

ప్రోగ్రాం రాసే విధానం

* దీన్ని సాల్వ్ చేయడానికి, మనం ఒక logic ఉపయోగిద్దాం.
(relational expression)*variable
* ఒక రిలేషనల్ ఆపరేటరు ఫలితం విలువ సత్యం (True) లేదా అసత్యం (False) అవుతుంది. దాన్నే సంఖ్యాపరంగా 1 లేదా 0 అని చెప్పవచ్చు
* ఇచ్చిన విలువ relational expression సత్యం అయితే మార్కులు వస్తాయి. అసత్యం అయితే 0 వస్తుంది. ఎందుకంటే relational expression ఉండే బ్రాకెట్ విలువ 1 లేదా 0 అవుతుంది.

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
    {
    int dec, n;
    printf("Enter a student marks\n");
    scanf("%d",&n);
    dec=(n>=35)*n;
    printf("Result=%d\n", dec);
```

```
return (0);
}
```

```
C:\lesson6.exe
Enter a student marks
30
Result=0
```

ఇచ్చిన మార్కులు 35 కంటే తక్కువగా ఉన్నాయి అంటే విద్యార్థి ఫెయిల్ అయ్యాడు.
అందుకే అవుట్‌పుట్ 0 వచ్చింది
ఇప్పుడు ఇన్‌పుట్‌గా 78 ఇవ్వాలి.
* రాసిన ప్రోగ్రాం ప్రకారం ఇచ్చిన మార్కులు 35 కంటే ఎక్కువగా ఉన్నాయి. కాబట్టి విద్యార్థి ఉత్తీర్ణత సాధించాడు అప్పుడు అవుట్ పుట్ 78 గా ప్రింట్ అవుతుంది.

```
C:\lesson6.exe
Enter a student marks
78
Result=78
```

ఉదాహరణ: 3

ఒక విద్యార్థి పరీక్ష మార్కులను ఇన్‌పుట్‌గా ఇవ్వాలి. పాస్ అవ్వడానికి కనీసం మార్కులు స్థిరంగా 35 కాకుండా, ఎంత ఇస్తే అంత కంటే ఎక్కువ వచ్చాయా లేదా లెక్కించి అతను పాస్ అయ్యాడో లేదో ప్రింట్ చేయాలి ?

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
    {
    int dec, pm, n;
    printf("Enter a student marks\n");
    scanf("%d",&n);
    printf("Enter minimum marks required to pass\n");
    scanf("%d", &pm);
    dec=(n>=pm)*n;
    printf("Result=%d\n", dec);
    return (0);
    }
```

Output

```
C:\lesson6.exe
Enter a student marks
30
Enter minimum marks required to pass
50
Result=0
```

```
C:\lesson6.exe
Enter a student marks
78
Enter minimum marks required to pass
50
Result=78
```

* ప్రోగ్రాంలో మనం పాస్ మార్కులు 50 అని నిర్ణయిస్తే అతడి మార్కులు 50 కంటే తక్కువగా వస్తే ఫెయిల్ ఎక్కువ వస్తే పాస్ అయినట్టు ప్రింట్ చేయాలి.
ఇటువంటి ప్రోగ్రాలు పోటీ పరీక్షల్లో కట్ఆఫ్ నిర్ణయించడానికి ఉపయోగిస్తారు
ఉదా: ఐబీపీఎస్ పరీక్షలో ప్రతి యేటా కట్ ఆఫ్ మార్కులు మారతాయి. పరీక్ష అనంతరం 100 కట్ఆఫ్ అని ఇస్తే అంతకంటే ఎక్కువగా ఉన్నవాళ్లు పాస్ అయితారు తక్కువ వచ్చిన వాళ్లు అర్హత పొందరు.

ఉదాహరణ: 4

ముగ్గురు విద్యార్థులు ఒక సబ్జెక్ట్‌లో సాధించిన మార్కులను ఇన్‌పుట్‌గా ఇచ్చి వారిలో ఎంత మంది పాస్ అయ్యారో లెక్కించి ప్రింట్ చేసే విధంగా ప్రోగ్రాం రాయండి ?

ప్రోగ్రాం రాసే విధానం:

* ఈ ప్రోగ్రాంలో ముందు ముగ్గురు విద్యార్థుల మార్కులు ఇన్‌పుట్ గా తీసుకోవాలి
* వారి మార్కులను విడివిడిగా 35 తో పోల్చిచూడాలి .
* 35 కంటే ఎక్కువ అయితే 1 తక్కువ అయితే 0 వచ్చే విధంగా ప్రోగ్రాం రాయాలి.
* వచ్చిన మూడు విలువలను కూడితే వచ్చే విలువను ప్రింట్ చేయాలి.

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
    {
    int a, b, c, np;
    printf("Enter three students marks\n");
    scanf("%d%d%d",&a,&b,&c);
    np=(a>=35) + (b>=35) +(c>=35) ;
    printf("Number of students passed=%d\n", np);
    return (0);
    }
```

```
C:\lesson6.exe
Enter three students marks
67   98   31
Number of students passed=2
```

ఉదాహరణ: 5

ముగ్గురు విద్యార్థులు ఒక సబ్జెక్ట్‌లో సాధించిన మార్కులను ఇన్‌పుట్‌గా ఇచ్చి వారిలో పాస్ అయిన వారి మార్కుల సగటు ప్రింట్ అయ్యేలా ప్రోగ్రాం రాయండి ?

గమనిక: పాస్ మార్కు 35

ప్రోగ్రాం రాసే విధానం

* ముందుగా ఎంత మంది విద్యార్థుల పాస్ అయ్యారో గుర్తించేలా ప్రోగ్రాం రాయాలి.
* ప్రతి విద్యార్థి మార్కున్ను లను 35 తో పోల్చాలి.
*ఎక్కువ అయితే 1 లేకపోతే 0 వస్తుంది. అంటే, పాస్ అయితే 1 లేకపోతే 0 లాగా తీసుకుంటాం. వీటిని కలిపితే ఎంత మంది పాస్ అయ్యారో వస్తుంది.
* ఉదాహరణ 2 లో రాసిన ప్రోగ్రాం logic ఉపయోగించుకొని పాస్ అయిన విద్యార్థుల మొత్తం మార్కులను కనుక్కోవాలి.
* అలా వచ్చిన వారి మార్కులను కలపాలి, తరువాత ఎంత మంది పాస్ అయ్యారో ఆ సంఖ్యతో మార్కులకు సరాసరి కనుక్కోవాలి

ఉదా ముగ్గురు విద్యార్థుల్లో ఇద్దరు పాస్ అయ్యారు. పాస్ అయిన విద్యార్థుల మార్కుల మొత్తం 150 అయితే 150/2 విలువను లెక్కించాలి

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
    {
    int a, b, c, np, sp;
    float average;
    printf("Enter three students marks\n");
    scanf("%d%d%d",&a,&b,&c);
    np=(a>=35) + (b>=35) +(c>=35) ;
    sp=(a>=35)*a + (b>=35)*b + (c>=35)*c;
    average=sp/(float)np;
    printf("Average marks of passed students=%f\n",
average);
    return (0);
    }
```

Output

```
C:\lesson6.exe
Enter three students marks
31 67 50
Average marks of  passed students=58.500000
```

ఇప్పుడు 31, 32, 29 ఇన్‌పుట్‌గా ఇవ్వండి . అవుట్‌పుట్ గమనించండి. అందరూ పెయిల్ అయ్యారు కాబట్టి భాగించే విలువ 0 అవుతుంది something / 0 విలువను కంప్యూటర్ లెక్కించలేదు.

```
C:\lesson6.exe
Enter three students marks
31 32 29
Average marks of  passed students=-1.#IND00
```

మీరు ఇంకో experiment చేయండి. పైన ఇచ్చిన ప్రోగ్రాంలో typecasting తీసివేసి యావరేజ్ కనుక్కోవడానికి ప్రయత్నించండి. కొన్ని compilers లో devided by zero అనే error రావచ్చు, కొన్నింటిలో మీ ప్రోగ్రామ్ hang అవ్వవచ్చు. ఎందుకంటే 0/0, కంప్యూటరు కూడా ఏమీ చెయ్యలేదు. ఇది మాథ్స్ లాగానే un-determined.

ఉదాహరణ: 6

ముగ్గురు విద్యార్థుల పరీక్ష మార్కులను ఇన్‌పుట్‌గా ఇవ్వాలి. పాస్ అవడానికి కనీసం మార్కులు స్థిరంగా 35 అనకుండా, ఎంత ఇస్తే అంత కంటే ఎక్కువ వచ్చాయో లేదో లెక్కించి అతడు అర్హత సాధించాడా లేదా గుర్తించాలి , పాస్ అయిన వారి మార్కుల సగటును ప్రింట్ చేయాలి ?

ప్రోగ్రాం రాసే విధానం

* ఉదాహరణ 1,2,3,4,5 అన్ని ప్రోగ్రాంలను పరిశీలించి ఈ ప్రోగ్రాం రాయండి

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
    {
    int a,b,c,np,sp,pm;
    float average;
    printf("Enter three students marks\n");
    scanf("%d%d%d",&a,&b,&c);
    printf("Enter minimum marks to pass\n");
    scanf("%d",&pm);
```

```
    np=(a>=pm) + (b>=pm) +(c>=pm) ;
    sp=(a>=pm)*a + (b>=pm)*b + (c>=pm)*c;
    average=sp/(float)np;
    printf("Average marks of passed students=%f\n",
average);
    return (0);
    }
```

Output

```
C:\lesson6.exe
Enter three students marks
39   67 89
Enter minimum marks to pass
50
Average marks of  passed students=78.000000
```

Conditional operator (షరతుల ఆపరేటర్లు)
దీన్నే compact if అని కూడా అంటారు. దీన్ని రెండు విధాలుగా ఉపయోగించవచ్చు.

1) (expr) ? expr1: expr2;
2)Variable=(expr)? expr1: expr2;

* మొదటి పద్ధతిలో , expr సత్యం (True) అయితే expr1 రన్ అవుతుంది, లేకపోతే expr2 రన్ అవుతుంది.
* రెండో పద్ధతిలో expr సత్యం అయితే expr1 రన్ అయి వచ్చిన విలువను ఇచ్చిన వేరియబుల్ కు assign చేస్తుంది లేకపోతే expr2 రన్అయి వచ్చిన విలువను ఇచ్చిన వేరియబుల్ కు assign చేస్తుంది.

ఉదాహరణ: 7

ఒక విద్యార్థి టెస్ట్ మార్కులు తీసుకొని అతడు పాస్ లేదా ఫెయిల్ అయినట్లు ప్రింట్ చేసే విధంగా ప్రోగ్రాం రాయండి ?

గమనిక: ఇక్కడ పాస్ మార్కులు 35

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
    {
    int n;
    printf("Enter a student marks\n");
    scanf("%d",&n);
    (n>=35)?printf("Passed\n"):printf("Failed\n");
    return (0);
    }
```

Output 1

ఇన్‌పుట్‌గా 70 ఇస్తే అవుట్‌పుట్

```
C:\lesson6.exe
Enter a student marks
70
Passed
```

Output 2

ఇన్‌పుట్‌గా 30 ఇస్తే అవుట్‌పుట్

```
C:\lesson6.exe
Enter a student marks
30
Failed
```

ఉదాహరణ: 8

ఒక విద్యార్థి టెస్ట్ మార్కులను ఇన్‌పుట్‌గా ఇచ్చి అతడు పాస్ అయ్యాడో లేదో ప్రింట్ చేయాలి. ఇక్కడ పాస్ అవడానికి కనీసం మార్కులు స్థిరంగా 35 కాకుండా, ఎంత ఇస్తే అంత కంటే ఎక్కువ వచ్చాయో లేదో లెక్కించి అతను పాస్ అయ్యాడో లేదో గుర్తించాలి ?

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
    {
    int n,pm;
    printf("Enter a student marks\n");
    scanf("%d",&n);
    printf("Enter minimum marks needed to pass\n");
    scanf("%d",&pm);
    (n>=pm)?printf("Passed\n"):printf("Failed\n");
    return (0);
    }
```

Output 1

ఇన్‌పుట్‌గా 70 ఇస్తే అవుట్‌పుట్

```
C:\lesson6.exe
Enter a student marks
70
Enter minimum marks needed to pass
50
```

Output 2

ఇన్‌పుట్‌గా 30 ఇస్తే అవుట్‌పుట్

```
C:\lesson6.exe
Enter a student marks
30
Enter minimum marks needed to pass
50
Failed
```

ఉదాహరణ: 9

ఒక విద్యార్థి ఐదు సబ్జెక్టుల మార్కులు తీసుకొని అతడు పాస్ అయ్యాడో లేదో ప్రింట్ చేయాలి. ఏ ఒక్క సబ్జెక్ట్‌లో ఫెయిల్ అయినా పరీక్షలో ఫెయిల్ అయినట్టు లెక్కించాలి. ?

గమనిక: సబ్జెక్ట్ పాస్ మార్కు 35

ప్రోగ్రాం రాసే విధానం

* ప్రోగ్రాంలో ప్రతి సబ్జెక్ట్ మార్కులను 35 తో పోల్చాలి.
* అంటే పాస్ అయితే 1 లేదా 0 అవుతుంది.
* పాస్ అయిన సబ్జెక్ట్‌లను కూడితే 5 వస్తే విద్యార్థి పరీక్షలో పాస్ అయినట్టుగా రావాలి లేదా ఫెయిల్ అని వచ్చేలా ప్రోగ్రాం రాయాలి.

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
    {
    int a, b, c, d, e, np;
    printf("Enter a student marks in five tests\n");
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    np=(a>=35) + (b>=35) + (c>=35) + (d>=35) +
(e>=35);
    (np==5)?printf("Passed\n"):printf("Failed\n");
    return (0);
    }
```

Output 1

ఇన్‌పుట్‌గా 67, 56, 39, 49, 99 ఇస్తే అవుట్‌పుట్

```
C:\lesson6.exe
Enter a student marks  in five tes
67 56 39 49 99
Passed
```

Output 2

ఇన్‌పుట్‌గా 89, 91, 31, 33, 56 ఇస్తే అవుట్‌పుట్

```
C:\lesson6.exe
Enter a student marks  in five
89 92 31 33 56
Failed
```

ఉదాహరణ: 10

ఒక విద్యార్థి ఐదు సబ్జెక్టుల మార్కులు తీసుకొని అతడు పాస్ అయ్యాడో లేదో ప్రింట్ చేయాలి. ఏ ఒక్క సబ్జెక్ట్‌లో ఫెయిల్ అయినా పరీక్షలో ఫెయిల్ అయినట్టు లెక్కించాలి. పాస్ అవ్వడానికి కనీసం మార్కులు స్థిరంగా 35 కాకుండా, ఎంత ఇస్తే అంత కంటే ఎక్కువ వచ్చాయో లేదో లెక్కించి అతడు పాస్ అయ్యాడో లేదో గుర్తించి ప్రింట్ చేయాలి

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
    {
    int a,b,c,d,e,np,pm;
    printf("Enter a student marks in five tests\n");
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    printf("Enter minimum marks required to pass a
test\n");
    scanf("%d",&pm);
    np=(a>=pm) + (b>=pm) + (c>=pm) + (d>=pm) +
(e>=pm);
    (np==5)?printf("Passed\n"):printf("Failed\n");
    return (0);
    }
```

Output

```
C:\lesson6.exe
Enter a student marks  in five tests
78 98 56 76 90
Enter minimum marks required to pass a test
50
Passed
```

```
C:\lesson6.exe
Enter a student marks  in five tests
78 92 34 77 49
Enter minimum marks required to pass a test
50
Failed
```

ఉదాహరణ: 11

ఒక విద్యార్థి ఐదు సబ్జెక్టుల మార్కులు తీసుకొని అతడు పాస్ అయ్యాడో లేదో ప్రింట్ చేయాలి. ఏ ఒక్క సబ్జెక్ట్‌లో ఫెయిల్ అయినా పరీక్షలో ఫెయిల్ అయినట్టు లెక్కించాలి. అంతే కాకుండా విద్యార్థి కి వచ్చిన మార్కులను బట్టి
First Class ,Second Class, Third Class, Failed , అని కూడా ప్రింట్ చేయాలి?

ప్రోగ్రాం రాసే విధానం

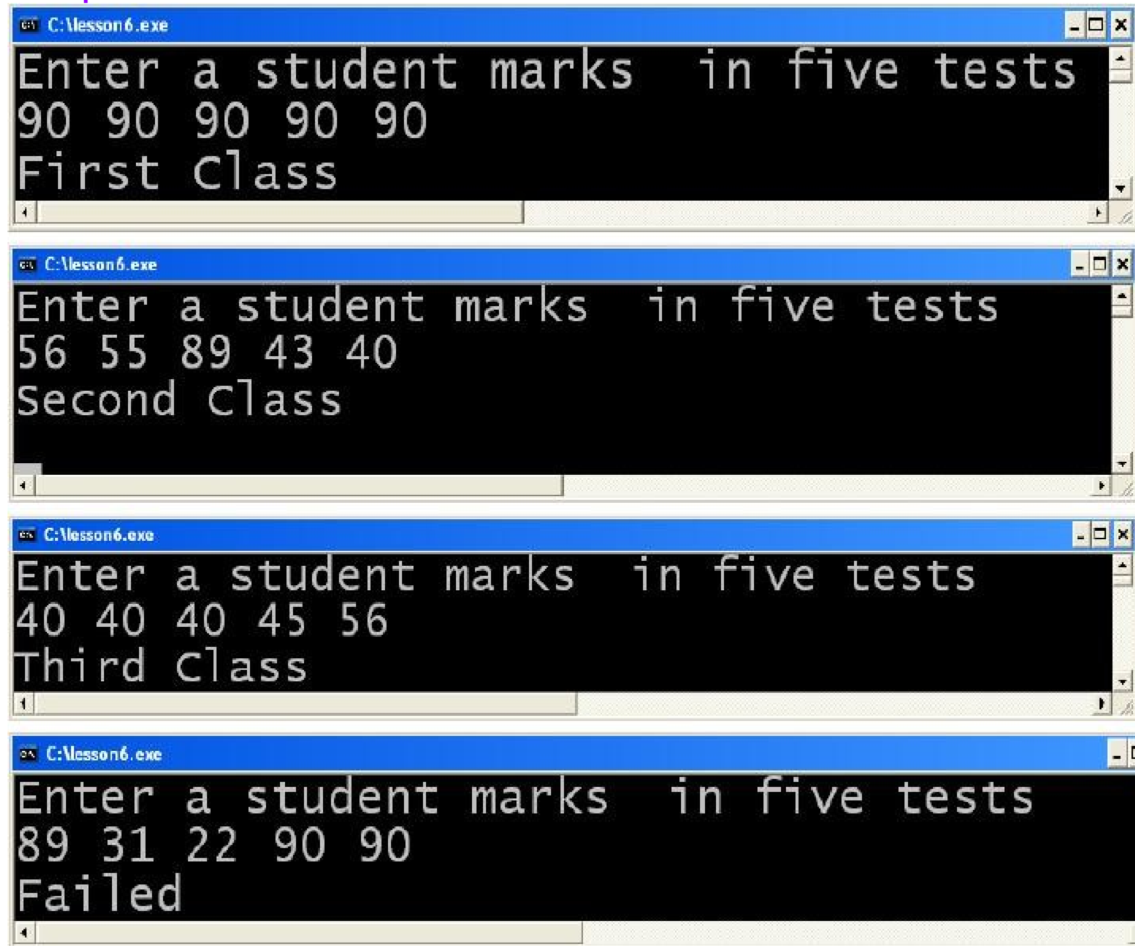
* ప్రోగ్రాంలో ప్రతి సబ్జెక్ట్ మార్కులను 35 తో పోల్చాలి.
* అంటే పాస్ అయితే 1 లేదా 0 అవుతుంది.
* పాస్ అయిన సబ్జెక్ట్‌లను కూడితే 5 వస్తే విద్యార్థి పరీక్షలో పాస్ అయినట్టుగా రావాలి లేదా ఫెయిల్ అని వచ్చేలా ప్రోగ్రాం రాయాలి.
* విద్యార్థి అన్ని పాస్ అయితే 300 కన్నా ఎక్కువ వస్తే ఫస్ట్ క్లాస్ అని, 250 కన్నా ఎక్కువ అయితే సెకండ్ క్లాస్ అని, లేకపోతే థర్డ్ క్లాస్ అని వచ్చేలా ప్రోగ్రాం రాయాలి.
* ఇలా రావడం కోసం ఈ ప్రోగ్రాంలో conditional if లోపల conditional if వాడాలి. ఇలా వాడటాన్ని nesting అని అంటారు.

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
    {
    int a,b,c,d,e,np,s;
    printf("Enter a student marks in five tests\n");
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    np=(a>=35) + (b>=35) + (c>=35) + (d>=35) +
(e>=35);
    (np==5)?((s=a+b+c+d+e)>=300?printf("First
Class\n"): ( (s>=250)? printf("Second
```

```
Class\n"):printf("Third Class\n") ) ):printf("Failed\n");
    return (0);
    }
```

Output



ఉదాహరణ: 12

ఒక విద్యార్థి ఐదు సబ్జెక్టుల మార్కులు తీసుకొని అతడు పాస్ అయ్యాడో లేదో ప్రింట్ చేయాలి. ఏ ఒక్క సబ్జెక్ట్‌లో ఫెయిల్ అయినా పరీక్షలో ఫెయిల్ అయినట్టు లెక్కించాలి. అంతే కాకుండా విద్యార్థి కి వచ్చిన మార్కులను బట్టి

Class 1
Class 2
Class 3
Failed

అని కూడా ప్రింట్ చేయాలి?

```
#include<stdio.h>
int main()
    {
    int a,b,c,d,e,np,s;
    printf("Enter a student marks in five tests\n");
```

```
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);

    np=(a>=35) + (b>=35) + (c>=35) + (d>=35) +
(e>=35);

    (np==5)?printf("Class %d\n", (4-
((s=a+b+c+d+e)>=300) - (s>=250)-(s>=175) )):
printf("Failed\n");
    return (0);
    }
```

ఈ ప్రోగ్రాం అవుట్‌పుట్‌ను మీరు రన్ చేసి కనుక్కోండి.

ఉదాహరణ: 13

ఐదుగురు విద్యార్థులు ఒక సబ్జెక్ట్‌లో సాధించిన మార్కులను ఇన్‌పుట్‌గా ఇచ్చి వారిలో పాస్ అయిన వారి మార్కులు సగటు ప్రింట్ అయ్యేలా ప్రోగ్రాం రాయండి ? దీనిలో ఏ run-time error లేకుండా రాయాలి.

ప్రోగ్రాం రాసే విధానం

* ఉదాహరణ 5 లో రాసిన ప్రోగ్రాంలో run-time error వచ్చింది. అది రాకుండా ప్రోగ్రాం రాయాలి
* something /0 విలువ రాకుండా జాగ్రత్త పడాలి. అందుకోసం
* సగటు లెక్కించేటప్పుడు కొంతమంది అయినా పాస్ అయిన వాళ్లు ఉంటేనే division జరిగేటట్లు కండీషన్ ఉపయోగించాలి. అలా అయితే అందరూ ఫెయిల్ అయితే division జరగదు అప్పుడు something /0 రాదు కాబట్టి run-time error ఉండదు.

(np విలువ 0 కాకపోతే, సగటు కాలిక్యులేట్ చేయాలి. np విలువ 0 కాకపోతే సత్యం అవుతుంది. అప్పుడు సగటు లెక్కించే విధంగా ప్రోగ్రాం రాయాలి.

ప్రోగ్రాం

```
#include<stdio.h>
int main()
    {
    int a,b,c,d,e,np,sp;
    printf("Enter Five students marks in a test\n");
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    np=(a>=35) + (b>=35) + (c>=35) + (d>=35) +
(e>=35);
    sp=(a>=35)*a + (b>=35)*b + (c>=35)*c +
(d>=35)*d + (e>=35)*e;
    (np)?printf("Passed students Average Marks=%f\n",
sp/(float)np):printf("All are failed!\n");
```

```
    return (0);
    }
```

Output

```
C:\lesson6.exe
Enter Five  students marks  in a test
79 90 88 23 21
Passed students Average Marks=85.666667
```

```
C:\lesson6.exe
Enter Five  students marks  in a test
12  31  23 32  19
All are failed!
```

ఉదాహరణ: 14

రెండు అంకెలను ఇన్‌పుట్ గా ఇస్తే ఆ రెండు అంకెలు ఒకే రేఖ పై ఉన్నాయో లేదో ప్రింట్ చేసేలా ప్రోగ్రాం రాయండి ?

ప్రోగ్రాం రాసే విధానం

* coordinate geometry కి సంబంధించిన ప్రోగ్రాం.
* ఒక లైన్ సాధారణ రూపం ax+by+c = 0 ఈ ఈక్వేషన్ లో రెండు పాయింట్లను పతక్షేపించాలి.దాని విలువ సున్నా వస్తే ఆ రెండు పాయింట్లు లైన్‌పై ఉన్నట్లు లేకపోతే లేదు అని వచ్చేలా ప్రోగ్రాం రాసుకోవాలి.
* x, y పాయింట్ లను ఇన్‌పుట్ గా ఇస్తాం
* తరువాత లైన్ ఈక్వేషన్ కి సంబంధించిన Parameters a,b,c గా ఇవ్వాలి.

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
    {
    float a,b,c,x,y;
    printf("Enter x and y co-ordinates of the point\n");
    scanf("%f%f",&x,&y);
    printf("Enter coefficients of line\n");
    scanf("%f%f%f",&a,&b,&c);
    ( a*x+b*y+c==0)? printf("On the given Line\n"):
printf("Not on the given Line\n");
    return (0);
    }
```

Output

```
C:\lesson6.exe
Enter x and y co-ordinates of the point
3 4
Enter coefficients of line
2 4 -3
Not on the given Line
```

```
C:\lesson6.exe
Enter x and y co-ordinates of the point
3  3
Enter coefficients of line
1  -1 0
On the given Line
```

ఉదాహరణ: 15

రెండు అంకెలను ఇన్‌పుట్ గా ఇస్తే ఆ రెండు అంకెలు ఒకే రేఖ పై ఉన్నాయో లేదో చెప్పటంతో పాటు దాని వాలు (slop) , ఇంటర్‌సెప్ట్ (intercept) లను కనుక్కునే విధంగా ప్రోగ్రాం రాయాలి?

ప్రోగ్రాం రాసే విధానం

* ఈ ప్రోగ్రాంలో వాలు , ఇంటర్సెప్ట్ కనుక్కోవాలి కాబట్టి coordinate geometry లోని y=mx+c సూత్రం ఉపయోగించుకోవాలి.

* ఇక్కడ పాయింట్లతో పాటు m, c parameters విలువలు ఇవ్వాలి.

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
    {
    float m,c,x,y;
    printf("Enter x and y co-ordinates of the point\n");
    scanf("%f%f",&x,&y);
    printf("Enter slope and intercept of line\n");
    scanf("%f%f",&m,&c);

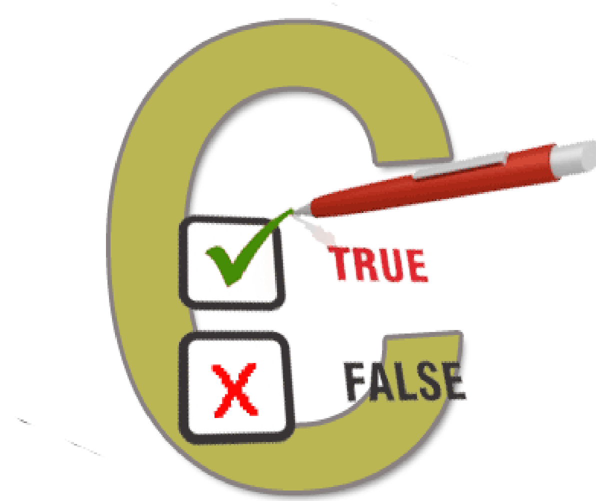
    ( y==(m*x+c))? printf("On the given Line\n"):
printf("Not on the given Line\n");
    return (0);
    }
```

Output

```
C:\lesson6.exe
Enter x and y co-ordinates of the point
3 4
Enter coefficients of line
2 4 -3
Not on the given Line
```

```
C:\lesson6.exe
Enter x and y co-ordinates of the point
3   3
Enter coefficients of line
1   -1 0
On the given Line
```

మరికొన్ని ' కండిషనల్ ' ప్రోగ్రామ్‌లు !



'8'లేషనల్, కండిషనల్ ఆపరేటర్లను ఉపయోగించి ప్రోగ్రాం రాయడం ఇంతకుముందు నేర్చుకున్నాం . ఈ పాఠంలో మరిన్ని ఉదాహరణలను అభ్యాసం చేద్దాం. అంతే కాకుండా మరికొన్ని ఆపరేటర్లను ఉపయోగించి ప్రోగ్రామ్‌లు రాయడం తెలుసుకుందాం.

ఉదాహరణ: 1

ఇన్‌పుట్‌గా ఒక పాయింట్ ఇచ్చి ఆ పాయింట్ వృత్తం పై ఉందో లేదో ప్రింట్ చేసేలా ప్రోగ్రాం రాయండి?

ప్రోగ్రాం రాసే విధానం

* ముందుగా ప్రోగ్రాంలో పాయింట్ x, y co-ordinates తో పాటు circle పెరామీటర్స్, వ్యాసార్థం ఇవ్వాలి.
* పాయింట్ వృత్తం పై ఉందో లేదో తెలియాలంటే అది వృత్త కేంద్రం నుంచి ఎంత దూరంలో ఉందో కనుక్కోవాలి
* ఆ దూరం వ్యాసార్థనికి సమానం అయితే పాయింట్ వృత్తంపై ఉన్నట్లుగా ప్రింట్ చేసే ప్రోగ్రాం రాయాలి

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
{
    float a,b,r,x,y;
    printf("Enter x and y co-ordinates of the point\n");
```

```
    scanf("%f%f",&x,&y);
    printf("Enter center co-ordinates of circle and its
radius\n");
    scanf("%f%f%f",&a,&b,&r);
    ( ((x-a)*(x-a)+(y-b)*(y-b))==r*r)? printf("On the
Circle\n"): printf("Not on the given circle\n");
    return (0);
}
```

Output:

```
C:\lesson7.exe
Enter x and y co-ordinates of the point
3 4
Enter center co-ordinates of circle and its
1 0 4
Not on the given circle
```

ఉదాహరణ: 2

మూడు సంఖ్యలను ఇన్ పుట్ గా ఇచ్చి ఆ మూడింటిలో పెద్ద సంఖ్యను గుర్తించి ప్రింట్ చేసే ప్రోగ్రాం రాయండి?

ప్రోగ్రాం రాసే విధానం

* ముందుగా మూడు సంఖ్యలను ఇన్ పుట్ గా తీసుకునేలా ప్రోగ్రాం రాయాలి
* తరువాత మూడో సంఖ్యను ఇతర సంఖ్యలతో పోలుస్తు conditional if ప్రోగ్రాం రాయాలి

ప్రోగ్రాం :

```
#include<stdio.h>
int main()
{
    int a,b,c,max;
    printf("Enter three integers\n");
    scanf("%d%d%d",&a,&b,&c);
    max=(a>b)?a:b;
    max=(c>max)?c:max;
    printf("Maximum out of three=%d\n", max);
    return 0;
}
```

Output:

```
C:\lesson7.exe
Enter three integers
78 98 33
Maximum out of three=98
```

{ పై ప్రోగ్రాంను మరో విధంగా కూడా రాయవచ్చు.
* మొదటి సంఖ్యను ముందుగా అన్నింటికన్నా పెద్దదిగా తీసుకోవాలి.
* రెండో సంఖ్యను రీడ్ చేసిన తర్వాత, అది మొదటి సంఖ్య కంటే పెద్దది అయితే రెండో సంఖ్యనే పెద్ద సంఖ్యగా కంప్యూటర్ గుర్తిస్తుంది
* ఇలానే మూడో సంఖ్య రెండో సంఖ్యను రీడ్ చేస్తుంది. ఇది పెద్ద సంఖ్య అయితే దీన్నే పెద్ద సంఖ్యగా కంప్యూటర్ గుర్తిస్తుంది. పెద్దది కాకపోతే రెండో సంఖ్య నే పెద్ద సంఖ్యగా ఉండిపోతుంది.
* ఇలా ఎన్ని రీడ్ చేసిన వాటికన్నా పెద్దదానికన్నా పెద్దది అయితే, దాన్నే అన్నింటికన్నా పెద్దది అని తీసుకుంటుంది. ఇలా ఎన్ని సంఖ్యలు ఉన్నఅందులో పెద్ద సంఖ్యను కనుక్కోవచ్చు
ఈ పద్ధతి గురించి తరువాతి పాఠంలో నేర్చుకుందాం. }

పై ప్రోగ్రాంలో ఇన్‌పుట్‌ని మరో విధంగా ప్రింట్ చేసేవిధంగా ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
    int a,max;
    printf("Enter first number\n");
    scanf("%d",&max);
    printf("Enter second number\n");
    scanf("%d",&a);
    max=(a>max)?a:max;
    printf("Enter Third number\n");
    scanf("%d",&a);
    max=(a>max)?a:max;
    printf("Maximum out of three=%d\n", max);
    return 0;
}
```

Output

```
C:\lesson7.exe
Enter first number
78
Enter second number
89
Enter Third number
33
Maximum out of three=89
```

ఉదాహరణ: 3

అసలు (principal amount) (p), శాతం (r), సమయం (t) సంవత్సరాల్లో ఇన్ పుట్‌గా తీసుకొని వడ్డీ లెక్కించి ప్రింట్ చేయాలి. సమయం ఒక సంవత్సరం కన్నా తక్కువ అయితే సాధారణ వడ్డీ (simple interest) లేకపోతే చక్ర వడ్డీ (compound interest) ప్రకారం వడ్డీ లెక్కించాలి ?

ప్రోగ్రాం రాసే విధానం

* సాధారణ, చక్రవడ్డీల సూత్రాలను ఉపయోగించాలి
* సమయంలో తేడా వస్తే లెక్కించే విధంగా conditional if ప్రోగ్రాం రాయాలి

సూత్రం:

$$Simple\ interest = \frac{prt}{100}$$

$$\text{Compound interest} = p * (1 + r/100)^t - p$$

ప్రోగ్రాం:

```
#include<stdio.h>
#include<math.h>
int main()
{
      float p,r,t,interest;
      printf("Enter principal amount, rate and time\n");
      scanf("%f%f%f",&p,&r,&t);
      interest=(t<1)? (p*r*t/100.0): ( p*pow(1+r/100.0,t)-
p);
      printf("Interest=%f\n", interest);
      return 0;
}
```

Output:

```
C:\lesson7.exe
Enter principal amount, rate and time
1000 12.5 0.5
Interest=62.500000
```

ఉదాహరణ: 4

ఒక వ్యక్తి సంవత్సర జీతం ఇన్‌పుట్ గా తీసుకొని, అతడు కట్టాల్సిన ఆదాయపు పన్ను ఎంతో లెక్కించి ప్రింట్ చేసే ప్రోగ్రాం రాయండి

ప్రోగ్రాం రాసే విధానం

* ముందు టాక్స్ 0 అనుకుందాం . తర్వాత జీతం నుంచి రూ.1,00,000 తీసివేసి మిగిలిన దానికి 10% చొప్పున కడతాం.
* జీతం 1,50,000 కన్నా ఎక్కువ వుంటే, ఎక్కువవున్న దానికి 10% కట్టి ఇంతకు ముందు దానికి కలుపుతాము.
* జీతం 2,50,000 కన్నా ఎక్కువ ఉంటే, ఎక్కువవున్న దానికి 10% కట్టి ఇంతకు ముందు దానికి కలుపుతాం.

ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
    float sal,tax=0;
    printf("Enter total yearly salary\n");
    scanf("%f",&sal);
    tax=(sal>=100000)?(sal-100000)*0.1:tax;
    tax=(sal>=150000)?tax+(sal-150000)*0.1:tax;
    tax=(sal>=250000)?tax+(sal-250000)*0.1:tax;
    printf("Income Tax Payable=%f\n",tax);
    return 0;
}
```

Output

```
C:\lesson7.exe
Enter total yearly salary
80000
Income Tax Payable=0.000000
```

```
C:\lesson7.exe
Enter total yearly salary
120000
Income Tax Payable=2000.000000
```

```
C:\lesson7.exe
Enter total yearly salary
200000
Income Tax Payable=15000.000000
```

```
C:\lesson7.exe
Enter total yearly salary
300000
Income Tax Payable=40000.000000
```

మరో పద్ధతి

```
#include<stdio.h>
int main()
{
	float sal,tax=0;
	printf("Enter total yearly salary\n");
	scanf("%f",&sal);

	tax=0.1*( (sal>=100000)*(sal-100000)+
	(sal>=150000)*(sal-150000) +
	(sal>=250000)*(sal-250000) );
	printf("Income Tax Payable=%f\n",tax);
	return 0;
}
```

* ప్రోగ్రాం రాసేటప్పుడు రెండు కండిషన్లను కలపాల్సి ఉంటుంది.
* అప్పుడు, *, + లు ఉపయోగించవచ్చు . ఉదాహరణకు, ఈ కింద ఇచ్చిన
* (expr1)*(expr2) గా కండిషన్లను కలిపితే రెండూ (expr1, expr2) సత్యం (True) అయినప్పుడు మాత్రమే ఫలితం సత్యం వస్తుంది .
* అంటే expr1, expr2 లు రిలేషనల్ expressions అయితే, రెండు సత్యం అయితేనే మొత్తం సత్యం అవుతుంది.
* (expr1)+(expr2) గా కండిషన్లను కలిపితే expr1, expr2 లలో ఏ ఒక్కటి సత్యం అయినా సత్యం అవుతుంది.

ఉదాహరణ: 5

మూడు సంఖ్యలను ఇన్‌పుట్ గా తీసుకొని అవి సహజక్రమం (natural order) లో ఉన్నాయా లేదో గుర్తించి ప్రింట్ చేయాలి ?

ప్రోగ్రాం రాసే విధానం

* Natural ఆర్డర్ అంటే ఉదాహరణ 7, 8, 9 ఇస్తే ఆర్డర్ లో ఉన్నట్లు, 7, 9, 10 ఇస్తే ఆర్డర్ లో లేని అంకెలు.
* మొదటి రెండు సంఖ్యల తేడా 1, తరువాతి రెండు సంఖ్యల తేడా 1 ఉందో లేదో లెక్కించాలి.
* ఉదాహరణకు a,b,c లు మూడు సంఖ్యలు అయితే (b-a) 1, (c-b) 1 గా ఉంటే natural ఆర్డర్ అని ప్రింట్ చేసేలా ప్రోగ్రాం రాయాలి.

ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
     int a, b, c;
     printf("Enter three integers\n");
     scanf("%d%d%d", &a,&b, &c);
     (((b-a)==1)*((c-b)==1))?printf("Strict natural
order\n"):printf("Not in strict natural order\n");
     return 0;
}
```

Output

```
C:\lesson7.exe
Enter three integers
7 8 9
Strict natural order
```

```
C:\lesson7.exe
Enter three integers
7 9  10
Not in strict natural order
```

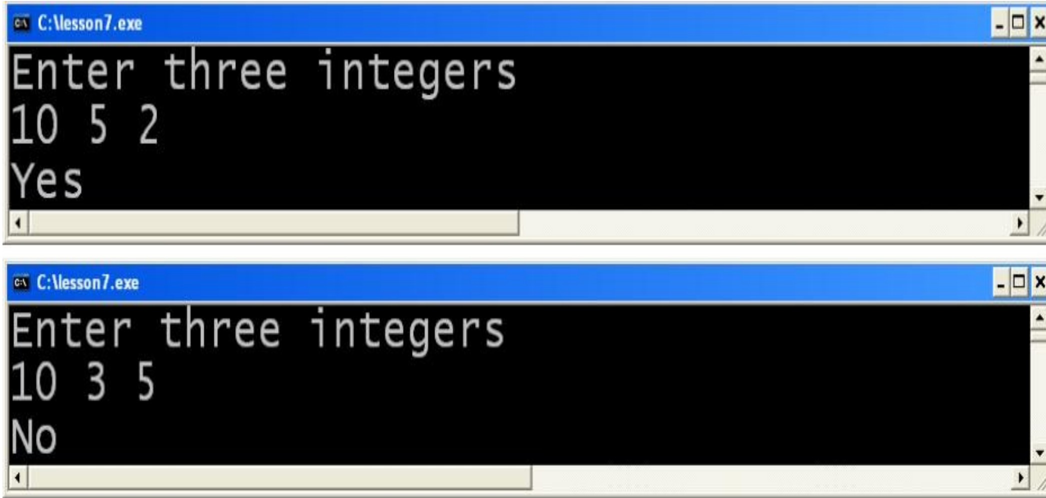
ఉదాహరణ 6

మూడు సంఖ్యలను ఇన్‌పుట్ గా ఇస్తే చివరి రెండు మొదటి సంఖ్యకు కారణాంకాలయితే అవును (Yes) అని లేకపోతే కాదు (No) అని ప్రింట్ చేయాలి ?

ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
      int a, b, c;
      printf("Enter three integers\n");
      scanf("%d%d%d", &a,&b, &c);
      ( ( a%b==0) * (a%c==0)) ?
printf("Yes\n"):printf("No\n");
      return 0;
}
```

Output



ఉదాహరణ 7

ఒక త్రిభుజం మూడు భుజాల కొలతలు తీసుకొని, అది సమబాహుత్రిభుజం (equilateral triangle) అవునా కాదా అని గుర్తించి ప్రింట్ చేయాలి ?

ప్రోగ్రాం రాసే విధానం

* త్రిభుజం మూడు భుజాలు సమానంగా ఉంటే దాన్ని సమబాహుత్రిభుజం అంటారు.
* a,b,c మూడు భుజాలనుకుంటే a=b=c అయితే దాన్ని సమబాహుత్రిభుజం అని ప్రింట్ చేయాలి.
* ఈ ప్రోగ్రాంలో మొదటి రెండు భుజాలను పోల్చి చూడాలి. తరువాత రెండు విలువలను పోల్చి చూడాలి.

ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
	float a, b, c;
	printf("Enter three sides of a traingle\n");
	scanf("%f%f%f", &a,&b, &c);
	( ( a==b) * (a==c)) ? printf("Yes\n"):printf("No\n");
	return 0;
}
```

Output

```
C:\lesson7.exe
Enter three sides of a traingle
7 7 7
Yes
```

```
C:\lesson7.exe
Enter three sides of a traingle
7 6 7
No
```

Implicit Assignment Statements

* Salary =Salary + bonus; అనే దానిని, షార్ట్ హ్యాండ్‌లో Salary += bonus; అని రాయొచ్చు.

ఇలా రాసే వాటినే implicit assignment statements అని అంటారు . ఇవి మన టైపింగ్ సమయాన్ని తగ్గిస్తాయి. పెద్ద ప్రోగ్రాంలు రాసేటప్పుడు ఈ పద్ధతి ఎక్కువగా ఉపయోగపడుతుంది.

a=a-10;	a-=10;
Xyz=Xyz*pqr;	Xyz*=pqr;
Xyz=Xyz/10;	Xyz/=pqr;
A=A%B;	A%=B;

ఉదాహరణ: 8

ఒక త్రిభుజం మూడు భుజాల కొలతలు తీసుకొని ఆ త్రిభుజం లంభకోణత్రిభుజం (Right angled triangle) అవునా కాదా అని గుర్తించి ప్రింట్ చేయాలి ?

ప్రోగ్రాం రాసే విధానం

* ఒక త్రిభుజం భుజాలు a, b, c లయితే ఏ రెండు భుజాల మొత్తం మూడోదాని వర్గం అయినా అది లంబకోణంత్రిభుజం అవుతుంది.

* ముందు combinations ను చెక్ చేయాలి. a^2+b^2= =c^2, a^2+c^2= =b^2, b^2+c^2= =a^2

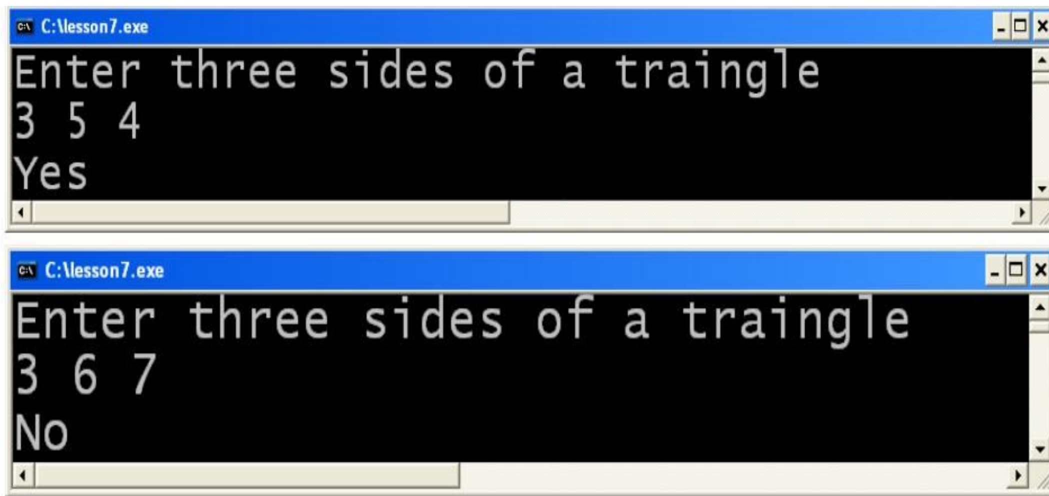
* ఇలా చేసేటప్పుడు రెండు రిలేషనల్ expression ల మధ్యలో + ను ఉపయోగిస్తాం . పైన చెప్పిన implicit assignment statements ఉపయోగించి వర్గా(square) లను లెక్కిస్తాం.

ప్రోగ్రాం:

```
#include<stdio.h>
int main()
{
```

```
    float a, b, c;
    printf("Enter three sides of a traingle\n");
    scanf("%f%f%f", &a,&b, &c);
    a*=a;
    b*=b;
    c*=c;
    ( ((a+b)==c) + ((a+c)==b) + ( (b+c)==a) ) ?
printf("Yes\n"):printf("No\n");
    return 0;
}
```

Output



ఉదాహరణ 9

ఒక సంవత్సరం నెంబర్ని ఇన్పుట్ గా ఇస్తే ఆ సంవత్సరం లీప్ సంవత్సరం అవునో కాదో అని లెక్కించి ప్రింట్ చేయాలి

ప్రోగ్రాం రాసే విధానం

* ఒక సంవత్సరం లీప్ సంవత్సరం అవునో కాదో తెలియాలంటే. దాన్ని 400 తో భాగించాలి శేషం సున్నా రావాలి అంతే కాకుండా ఆ సంఖ్య 100 తో నిశ్శేషంగా భాగించకూడదు

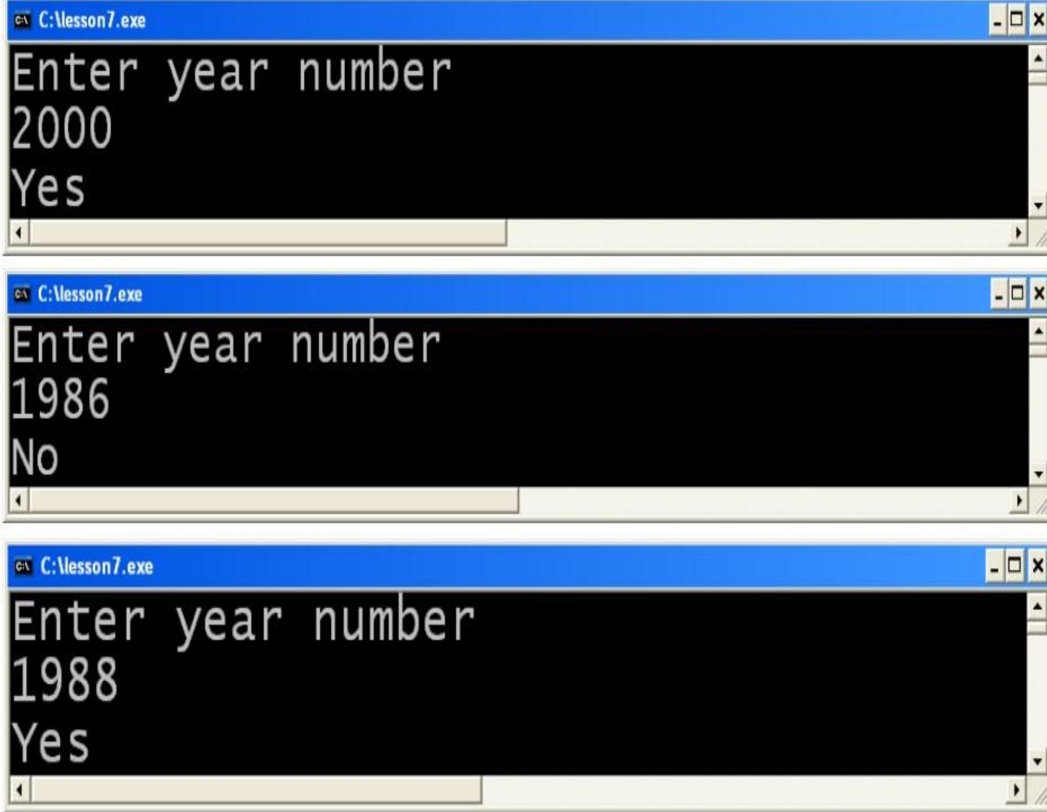
* మొదట రెండు లాజికల్ expressions మధ్యలో + ఉపయోగించాలి. అంతే కాకుండా అందులో ఉండే రెండు లాజికల్ expressions మధ్యలో * వాడాలి.

ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
    int y;
    printf("Enter year number\n");
    scanf("%d", &y);
```

```
	( ( y%400==0) + ( (y%4==0) * ( y%100!=0) ) ) ?
printf("Yes\n"):printf("No\n");
	return 0;
}
```

Output



ఉదాహరణ 10

మూడు సంఖ్యలను ఇన్‌పుట్‌గా ఇస్తే అందులో పెద్ద సంఖ్యను లెక్కించి ప్రింట్ చేసే ప్రోగ్రాం రాయండి ?

ప్రోగ్రాం రాసే విధానం

* ముందుగా ఇచ్చిన నెంబర్‌లలో మొదటి సంఖ్యను మిగిలిన సంఖ్యలతో పోల్చుతాం మొదటి సంఖ్య పెద్దది అయితే అది ప్రింట్ చేస్తాం కాకపోతే మిగిలిన వాటిని పోల్చుతాం ఇలా చెక్ చేసే విధంగా ప్రోగ్రాం రాయాలి

ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
	int a,b,c;
	printf("Enter three numbers\n");
	scanf("%d%d%d", &a,&b,&c);
	( (a>b)*(a>c) ) ? printf("%d\n",a):printf("%d",(b>c?
```

```
b:c) );
      return 0;
}
```

Output



పై ప్రోగ్రాంను మరో విధంగా రాసే పద్ధతి

```
#include<stdio.h>
int main()
{
      int a,b,c;
      printf("Enter three numbers\n");
      scanf("%d%d%d", &a,&b,&c);
      ( (a>b) ) ? printf("%d\n",(a>c? a: c) ):printf("%d",
(b>c? b: c) );
      return 0;
}
```

ఉదాహరణ 11

ఒక విద్యార్థి ఐదు సబ్జెక్టుల మార్కులు ఇన్‌పుట్‌గా తీసుకొని అతడు ఉత్తీర్ణుడయ్యాడో లేదో గుర్తించి ప్రింట్ చేయాలి

ప్రోగ్రాం రాసే విధానం

* ఈ ప్రోగ్రాంలో పాస్ మార్కులు రావాలంటే ప్రతి సబ్జెక్టులో 35 కన్నా ఎక్కువ రావాలి.
* ఈ లాజిక్‌ని * ఉపయోగించి రాసే పద్ధతి.

ప్రోగ్రాం

```
#include<stdio.h>
int main()
{
      int a,b,c,d,e,np,s;
      printf("Enter a student marks in five tests\n");
      scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
      ((a>=35)* (b>=35)*(c>=35)*( d>=35) *(e>=35) )?
printf("Passed\n"):printf("Failed\n");
      return 0;
}
```

Output

```
C:\lesson7.exe
Enter a student marks  in five tests
78 99 77 39 89
Passed
```

```
C:\lesson7.exe
Enter a student marks  in five tests
78 99 23 89 33
Failed
```

లాజికల్ ఆపరేటర్లను ఎలా ఉపయోగించాలి?

మనం ఇంతకు ముందు పాఠంలో 'సీ' లాంగ్వేజీలో చాలా ఆపరేటర్లను ఉపయోగించి ప్రోగ్రామ్ లు రాయడం నేర్చుకున్నాం. ఇప్పుడు లాజికల్ ఆపరేటర్ల గురించి తెలుసుకొని వాటిని ప్రాక్టికల్ గా ప్రోగ్రామ్ లో ఎలా ఉపయోగించాలో తెలుసుకుందాం.

'సీ' లాంగ్వేజీ లో రెండు లాజికల్ ఆపరేటర్లు ఉన్నాయి. అవి, లాజికల్ AND, లాజికల్ OR. వాటి సింబల్స్ &&, ||. ఇవి రెండూ కూడా బైనరీ ఆపరేటర్లు. అంటే వాటికి రెండు ఆపరాండ్స్ ఉంటాయి. అవి, వేరియబుల్స్ అయినా, కాన్ స్టెంట్ లు అయినా, e expressions అయినా కావచ్చు; ఏ టైపువి అంటే integer, float అయినా కావచ్చు.

రెండూ ఆపరాండ్ లు ట్రూ అయితే లాజికల్ AND ఫలితం ట్రూ (సత్యం) అవుతుంది. లేకపోతే ఫాల్స్ (అసత్యం) అవుతుంది. అలాగే, రెండూ ఆపరాండ్ లలో ఏ ఒక్కటి అయినా ట్రూ అయితే లాజికల్ OR ఫలితం ట్రూ అవుతుంది. లేకపోతే ఫాల్స్ అవుతుంది.

II ఈ క్రింది టేబుల్ చూడండి. ఇక్కడ, A, B లు రె0డు ఆపరాండ్ లు

<table>
<tr><td>A</td><td>B</td><td>A&&B</td><td>A||B</td></tr>
<tr><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>F</td><td>T</td><td>F</td><td>T</td></tr>
<tr><td>T</td><td>F</td><td>F</td><td>T</td></tr>
<tr><td>T</td><td>T</td><td>T</td><td>T</td></tr>
</table>

ఇక్కడ మనం ఇంతకు ముందు తెలుసుకున్న విషయాన్ని గుర్తు తెచ్చుకోవాలి. ఏ పాజిటివ్ వాల్యూ అయినా నెగటివ్ వాల్యూ అయినా ట్రూ లాగా తీసుకుంటుంది, సున్నా అంటే ఫాల్స్ లాగా తీసుకుంటుంది.

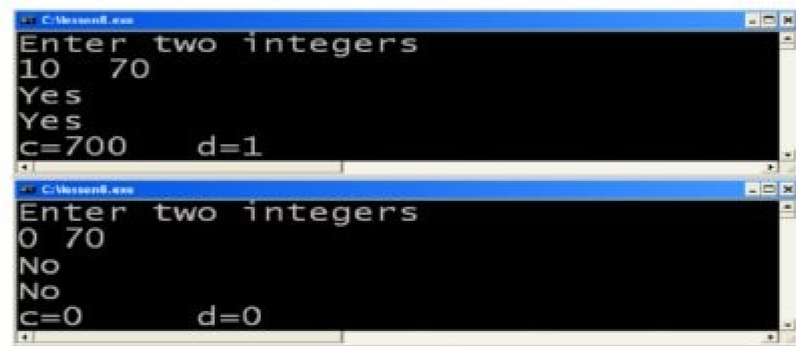
ముందు పాఠంలో, మనం రెండు కండిషనులు జాయిన్ చేయడానికి *, + లు ఉపయోగించి రాశాం. ఇప్పుడు వాటి బదులు లాజికల్ AND, లాజికల్ OR లు వాడవచ్చు. నిజానికి, లాజికల్ ఎక్స్ ప్రెష్ నులలో లాజికల్ AND, లాజికల్ OR లు *,

+ వాడవచ్చు. కానీ, arithmetic ఎక్స్ ప్రెష్ నులలో *, + la లలో లాజికల్ AND, లాజికల్ OR లు వాడలేం.

ఉదాహరణ 1

ఈ కింది ప్రోగ్రామ్, పైన చెప్పిన దానిని ప్రదర్శిస్తుంది. మనం, కింద ఇచ్చిన స్క్రీన్ లను కూడా గమనించవచ్చు. మనం, 10, 70 ఇస్తే *, లాజికల్ AND, రెండూ కూడా Yes అని ప్రింట్ చేశాయి. అదే, 0, 70 *, లాజికల్ AND, రెండూ కూడా No అని ప్రింట్ చేశాయి. అంటే, రెం0 డూ లాజికల్ గా ఒకే లాగా పని చేస్తున్నాయి. కానీ, c, d వాల్యూస్ చూస్తే ఒకటి కాదు. అంటే, రెండూ arithmetic గా ఒకటి కాదు.

```
#include<stdio.h>
int main()
{
    int a,b,c,d;
    printf("Enter two integers\n");
    scanf("%d%d",&a,&b);
    (a&&b )?printf("Yes\n"):printf("No\n");
    (a*b)?printf("Yes\n"):printf("No\n");
    c=a*b;
    d=a&&b;
    printf("c=%d\td=%d\n",c,d);
  return 0;
}
```



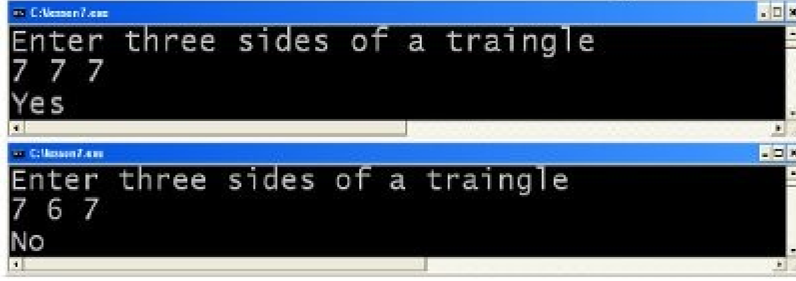
ఉదాహరణ 2

ఒక triangle మూడు భుజాల కొలతలు తీసుకొని, అది equilateral triangle అవునా కాదా అని ప్రింట్ చేయడానికి ప్రోగ్రాం ఇది. మనం a,b,c లను మూడు భుజాలు అని అనుకుంటే, మూడు ఒకటే అయితే, Yes అని రావాలి, లేకపోతే No అని రావాలి. మనం, మాథ్స్ లో a=b=c, కాబట్టి equilateral triangle అని రాస్తాం. కానీ, ఇక్కడ మనం మొదటి రెండూ ఒకటేనా అని, తరువాతి రెండూ ఒకటేనా అని చూస్తాం. దీనినే, ఇంకో విధంగా కూడా చేయవచ్చు. అది ముందు పాఠంలో చేశాం. ఇక్కడ లాజికల్ AND వాడి చేస్తున్నాం. ఈ కింది ప్రోగ్రాం పైన చెప్పిన విధంగా పని చేస్తుంది.

```
#include<stdio.h>
int main()
```

```
{
     float a, b, c;
     printf("Enter three sides of a traingle\n");
     scanf("%f%f%f", &a,&b, &c);
     ( ( a==b) && (a==c)) ? printf("Yes\n"):printf("No\n");
   return 0;
}
```

పై ప్రోగ్రాంను రన్ చేస్తూ వివిధ రకాలైన ఇన్ పుట్ లు ఇస్తే ఏమవుతుందో ఈ కింది స్క్రీన్ లు చూపిస్తాయి.



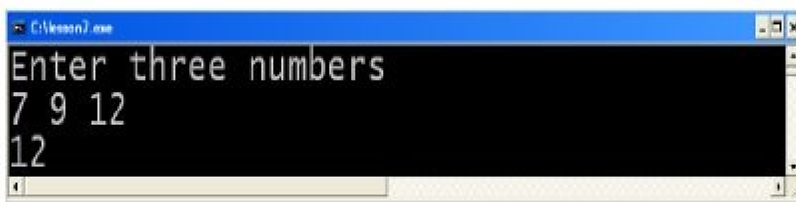
ఉదాహరణ 3

మూడు integers ను ఇన్ పుట్ లాగా తీసుకొని వాటిలో maximum వాల్యూను ప్రింట్ చేయడానికి ఈ ప్రోగ్రాం రాశాం. దీనిని ఇంతకు ముందే సాల్వ్ చేశాం. ఇప్పుడు ఇంకో రకంగా చేద్దాం.

ముందు, మొదటి నంబరు మిగతా రెండింటికన్నా పెద్దదా కాదా అని చెక్ చేస్తాం. పెద్దది అయితే, దానినే మూడింటిలోకెల్లా పెద్దది అని అంటాం. లేకపోతే, మిగతా రెండింటిలో పెద్దదాన్నే మూడింటిలోకెల్లా పెద్దది అని అంటాం.

```
#include<stdio.h>
int main()
{
     int a,b,c;
     printf("Enter three numbers\n");
     scanf("%d%d%d", &a,&b,&c);
     ( (a>b)&& (a>c) ) ? printf("%d\n",a):printf("%d",(b>c?
b:c) );
   return 0;
}
```

పై ప్రోగ్రాంను రన్ చేస్తూ ఇన్ పుట్ ఇస్తే ఏమవుతుందో ఈ కింది స్క్రీన్ చూపిస్తుంది.



ఉదాహరణ 4

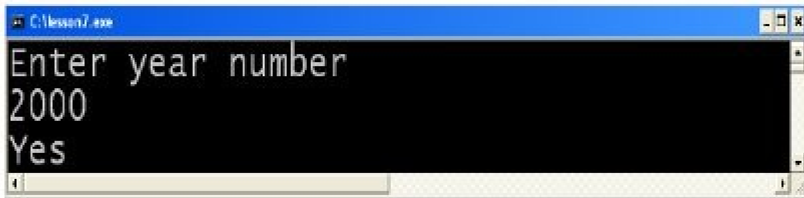
ఈ ప్రోగ్రాం, సంవత్సరం నంబర్ ను ఇన్ పుట్ లాగా తీసుకొని అది లీప్ సంవత్సరం అవునా కాదా అని చెబుతుంది. ఒక సంవత్సరం లీప్ సంవత్సరం కావాలి అంటే, ఈ కింది వాటిలో ఏదో ఒకటి ట్రూ (సత్యం) కావాలి.

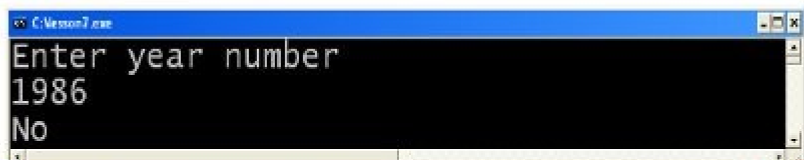
1. 400 తో ఆ ఇన్ పుట్ divide కావాలి.
2. 4 తో అది divide కావాలి, ఇంకా 100 తో divide కాకూడదు.

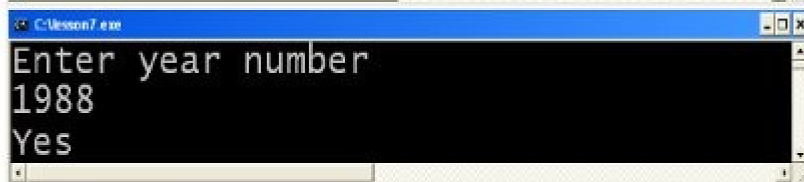
దీనిని మనం ముందు పాఠంలో కూడా చేశాం. మనం మొదట రెండు లాజికల్ expressions మధ్యలో OR వాడాలి. అలాగే, రెండో దానిలో ఉండే రెండు లాజికల్ expressions మధ్యలో AND వాడాలి. ఈ కింద పూర్తి ప్రోగ్రాం ఇచ్చాం.

```
#include<stdio.h>
int main()
{
     int y;
     printf("Enter year number\n");
     scanf("%d", &y);
     ( ( y%400==0) || ( (y%4==0) && ( y%100!=0) ) ) ?
printf("Yes\n"):printf("No\n");
   return 0;
}
```

పై ప్రోగ్రాంను రన్ చేస్తూ వివిధ రకాలైన ఇన్ పుట్ లు ఇస్తే ఏమవుతుందో ఈ స్క్రీన్ లు చూపిస్తాయి.







ఉదాహరణ 5

ఈ ప్రోగ్రాంలో ఒక విద్యార్థికి సంబంధించిన అయిదు పరీక్షల మార్కులను ఇన్ పుట్ గా తీసుకొని అతడు పాసయ్యాడా లేదా ఫెయిల్ అయ్యాడా చెప్పాలి.

పాస్ అవ్వాలి అంటే, ప్రతి టెస్ట్ లో 35 రావాలి. దీనిని మనం యింతకు ముందు పాఠంలో చేశాం. ఇప్పుడు దీనిని ఇంకో రకంగా చేస్తున్నాం. మొత్తం మీద పాస్ అవ్వాలి అంటే, మొదటి టెస్ట్ లో 35 కన్నా ఎక్కువ రావాలి, రెండో దానిలో కూడా 35 కన్నా ఎక్కువ రావాలి. అలాగే ప్రతి టెస్ట్ లో 35 కన్నా ఎక్కువ రావాలి. దానినే, && వాడి కింద విధంగా రాశాం. అంటే, ప్రతి టెస్ట్ లో 35 వచ్చాయా లేదా అని చూసే లాజికల్ expressions మధ్యలో && వాడాం.

```
#include<stdio.h>
int main()
```

```
{
     int a,b,c,d,e;
     printf("Enter a student marks in five tests\n");
     scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
     ((a>=35) && (b>=35) && (c>=35) && ( d>=35) &&
(e>=35) )? printf("Passed\n"): printf("Failed\n");
   return 0;
}
```

పై ప్రోగ్రాంను రన్ చేస్తూ వివిధ రకాలైన ఇన్ పుట్ లు ఇస్తే ఏమవుతుందో ఈ కింది స్క్రీన్ లు చూపిస్తాయి.



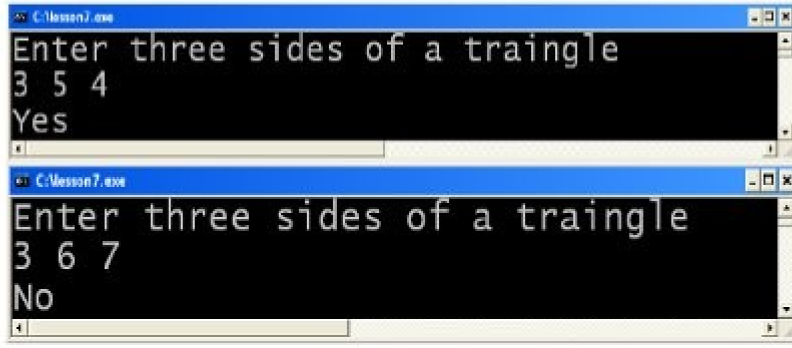


ఉదాహరణ 6

ఒక triangle మూడు భుజాల కొలతలు తీసుకొని, అది right angled triangle అవునా కాదా అని ప్రింట్ చేసే ప్రోగ్రాం ఇది. మనం మూడు భుజాలను a, b, c లు అని అనుకొంటే, E ఏ రెండూ భుజాల squares మొత్తం మూడో దాని square కు సమానం అయితే, అది right angled triangle అవుతుంది. అంటే, మనం మూడు combinations ను చెక్ చేయాలి. అవి, a2+b2==c2, a2+c2==b2, b2+c2==a2. ఇలా చేసేటప్పుడు రెండు రిలేషనల్ ఎక్స్‌ప్రెషనుల మధ్యలో || పెడతాం. ఈ ప్రోగ్రాంను ముందు పాఠంలో ఇంకో విధంగా + వాడి చేసాం.

```
#include<stdio.h>
int main()
{
     float a, b, c;
     printf("Enter three sides of a traingle\n");
     scanf("%f%f%f", &a,&b, &c);
     a*=a;
     b*=b;
     c*=c;
     ( ((a+b)==c) || ((a+c)==b) || ( (b+c)==a) ) ?
printf("Yes\n"):printf("No\n");
   return 0;
}
```

పై ప్రోగ్రాంను రన్ చేస్తూ వివిధ రకాలైన ఇన్ పుట్ లు ఇస్తే ఏమవుతుందో ఈ కింది స్క్రీన్ లు చూపిస్తాయి.



లాజికల్ AND ను ఎవాల్యుయేట్ చేస్తున్నప్పుడు, మొదటి ఆపరాండ్ ఫాల్స్ అయితే రెండో దానిని ఎవాల్యుయేట్ చేయదు. ఎందుకంటే, రెండోది ట్రూ అయినా ఫాల్స్ అయినా ఫైనల్ రిజల్ట్ ఫాల్స్ కాబట్టి. అలాగే, లాజికల్ OR ను ఎవాల్యుయేట్ చేస్తున్నప్పుడు, మొదటి ఆపరాండ్ ట్రూ అయితే రెండో దానిని ఎవాల్యుయేట్ చేయదు. ఎందుకంటే, రెండోది ట్రూ అయినా ఫాల్స్ అయినా ఫైనల్ రిజల్ట్ ట్రూ కాబట్టి.

ఉదాహరణ 7

పైన చెప్పిన పాయింట్ ను ప్రాక్టికల్ గా చూపించడానికి ఈ ప్రోగ్రాం ఇచ్చాం. ఈ ప్రోగ్రాంలో కూడా, ఒక విద్యార్థికి సంబంధించిన అయిదు సబ్జెక్టుల మార్కులు తీసుకొని, అతడు పాసయ్యాడా లేదా ఫెయిల్ అయ్యాడా చెప్పాలి. అంతే కాకుండా అతడి మొత్తం మార్కులు ప్రింట్ చేయాలి. ఇక్కడ కూడా పాస్ అవ్వాలి అ0టే, ప్రతి టెస్ట్ లో 35 మార్కులు రావాలి. మొత్తం మీద పాస్ అవ్వాలి అంటే, మొదటి టెస్ట్ లో 35 కన్నా ఎక్కువ రావాలి, రెండో దానిలో కూడా 35 కన్నా ఎక్కువ రావాలి. అలాగే ప్రతి టెస్ట్ లో 35 కన్నా ఎక్కువ రావాలి. దానినే, && వాడి పై ఉదాహరణలాగానే కింది విధంగా రాశాం. అంటే, ప్రతి టెస్ట్ లో 35 వచ్చాయా అని చూసే లాజికల్ expressions మధ్యలో && వాడాం. ఆఖరున (s=a+b+c+d+e) అనే దానిని కూడా వాడాం. దీన్ని ఎప్పుడు ఎవాల్యూయేట్ చేస్తుందంటే, ముందు ఉన్న అన్ని లాజికల్ AND ఆపరేటర్లు ట్రూ అయితే. లేకపోతే S వాల్యూ ముందుగా ఇచ్చిన సున్నా లాగా ఉంటుంది.

```
#include<stdio.h>
int main()
{
    int a,b,c,d,e,s=0;
    printf("Enter a student marks in five tests\n");
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    ((a>=35) && (b>=35) && (c>=35) && ( d>=35) &&
(e>=35)&&(s=a+b+c+d+e) )? printf("Passed\n"):
printf("Failed\n");
    printf("Total=%d\n",s);
  return 0;
}
```

ఈ స్క్రీన్ మనం మార్కులు 80, 90, 90, 90, 80 ఇస్తే ఏమవుతుందో చూపిస్తుంది. అన్నీ 35 కన్నా ఎక్కువ కాబట్టి, అంటే లాజికల్ AND లు అన్నీ ట్రూలు. కాబట్టి మొత్తాన్ని లెక్కించి ప్రింట్ చేస్తుంది.

```
C:\lesson8.exe
Enter a student marks  in five tests
80 90 90 90 80
Passed
Total=430
```

అదే మనం, 90, 80, 30, 70, 90 ఇస్తే టోటల్ సున్నా ఇస్తుంది. ఎందుకంటే, మూడో లాజికల్ AND ఫాల్స్ కాబట్టి. తర్వాత ఉండే వాటిని ఎవాల్యూయేట్ చేయదు.

```
C:\lesson8.exe
Enter a student marks  in five tests
90 80 30 70 90
Failed
Total=0
```

```
#include<stdio.h>
int main()
{
    int a,b,c,d,e,s=0;
    printf("Enter a student marks in five tests\n");
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    ((a>=35) && (b>=35) && (c>=35) && ( d>=35) &&
(e>=35)&&(s=a+b+c+d+e) )? printf("Class %d\n", (4-
(s>=360)-(s>=250)-(s>=175) ) ): printf("Failed\n");
  return 0;
}
```

ఉదాహరణ 8

ఈ ప్రోగ్రాంలో కూడా ఒక విద్యార్థి కి సంబంధించిన అయిదు సబ్జెక్టుల మార్కులు తీసుకొని, పాస్ అయితే ఏ క్లాసో ప్రింట్ చేయాలి. ఈ కింది రూల్స్ ఫాలో అవ్వాలి. ఇక్కడ కూడా పాస్ అవ్వాలి అంటే, ప్రతీ టెస్ట్ లో కనీసం 35 మార్కులు రావాలి.

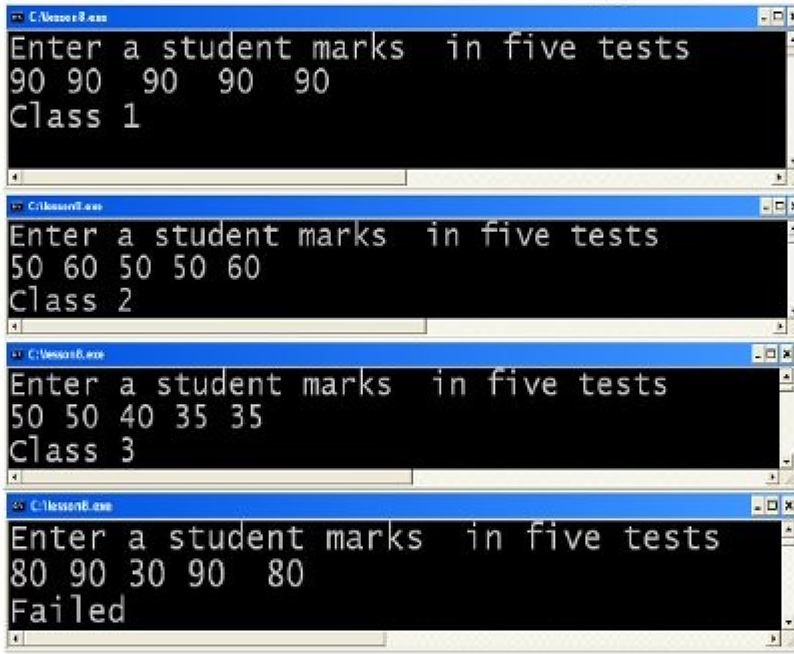
యావరేజ్ >=60 లేదా టోటల్ >=300	Class 1
యావరేజ్ >=50 లేదా టోటల్ >=250	Class 1
యావరేజ్ >=60 లేదా టోటల్ >=175	Class 1

ఈ కింది ప్రోగ్రాం, మనకు కావాల్సిన రిజల్ట్ ఇస్తుంది. ఇక్కడ పైన వాడిన లాజిక్ వాడాం. అదనంగా, ఏ క్లాసో చెప్పడానికి ఇంకో లాజిక్ వాడాం. మీరు ఆనాలిసిస్ చేస్తే ఇది ఎలా పని చేస్తుందో తెలుస్తుంది.

```
#include<stdio.h>
int main()
```

```
{
    int a,b,c,d,e,s=0;
    printf("Enter a student marks in five tests\n");
    scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
    ((a>=35) && (b>=35) && (c>=35) && ( d>=35) &&
(e>=35)&&(s=a+b+c+d+e) )? printf("Class %d\n", (4-
(s>=300)-(s>=250)-(s>=175) ) ): printf("Failed\n");
  return 0;
}
```

ఈ కింది స్క్రీన్ లు ఈ ప్రోగ్రాం ఎలా పని చేస్తుందో చూపిస్తాయి. ఇచ్చిన వాల్యూస్ అన్నీ 35 కన్నా ఎక్కువ కాబట్టి మొత్తం 450 అవుతుంది. అప్పుడు, (4-(s>=300)-(s>=250)-(s>=175)) అనేది (4- (450>=300) –(450>=250) –(4>=175)) లాగా అవుతుంది. అది, (4-1-1-1)=1 లాగా అవుతుంది. కాబట్టి మనకు, Class 1 అని స్క్రీన్ మీద వస్తుంది.



దీంతో ఈ పాఠం ద్వారా లాజికల్ AND, OR అనే ఆపరేటర్లను వాడి ప్రోగ్రాంలు రాయడం నేర్చుకున్నాం.

అవునంటే కాదనిలే.... కాదంటే అవుననిలే

* యూనరీ ఆపరేటర్లు



మనం ఇప్పటి వరకు నేర్చుకున్నవన్నీ బైనరీ, టెర్నరీ ఆపరేటర్లు. ఈ పాఠంలో 'సీ' లాంగ్వేజీలో ఉన్న యూనరీ ఆపరేటర్ల గురించి నేర్చుకుందాం. యూనరీ ఆపరేటర్లకు ఒక్క ఆపరాండ్ మాత్రమే వుంటుంది.

ఉదాహరణకు, c=a+b అనే దానిలో + కు రెండు ఆపరాండ్లు ఉన్నాయి, కాబట్టి దానిని బైనరీ + అని అంటారు. అదే, c= +b అనే దానిలో + ను యూనరీ + అని అంటారు. ఎందుకంటే దానికి ఒక్క ఆపరాండ్ మాత్రమే వుంది. అలాగే, c=-b అనే దానిలో – ను యూనరీ – అని అంటారు.
'సీ' లాంగ్వేజీలో negation, unary increment, decrement అనే మరికొన్ని యూనరీ ఆపరేటర్లు ఉన్నాయి. వాటి గురించి ఇప్పుడు తెలుసుకుందాం.

Negation Operator(!)

దీనిని మనం ఓ వేరియబుల్ కు అయినా, constant కు అయినా, expression కు అయినా అప్లై చేయవచ్చు. అంటే దీని ఆపరాండ్ వేరియబుల్ అయినా, constant అయినా, expression కు అయినా కావచ్చు. ఆపరాండ్ ట్రూ అయితే, దీనిని అప్లై చేసిన తర్వాత ఫాల్స్ అవుతుంది. అదే ఆపరాండ్ ఫాల్స్ అయితే, దీనిని అప్లై ఛేసిన తర్వాత రిజల్ట్ ట్రూ అవుతుంది. అంటే అవునంటే కాదనిలే, కాదంటే అవుననిలే అన్నమాట.
'సీ' లాంగ్వేజీలో positive అయినా, negative అయినా ట్రూ లాగా తీసుకొంటుంది. సున్నా అయితే ఫాల్స్ లాగా తీసుకుంటుంది. అలాగే, మనం ఇంతకు ముందు పాఠాల్లో ట్రూ అంటే 1 అని, ఫాల్స్ అయితే 0 అనీ తెలుసుకున్నాం. ఈ కింది ప్రోగ్రాంలో ఈ విషయాలను ఉపయోగించుకుందాం.

ఉదాహరణ

ఈ కింది ప్రోగ్రాం negation ఆపరేటరు వాడితే ఏమవుతుందో తెలియజేస్తుంది... చూడండి.

```
#include<stdio.h>
int main()
{
    int a=10, b=10, c=0, i,j,k,l,m,n,p,q,r;

    i=!a;
    j=!b;
    k=!c;
    l=!100;
    m=!-100;
    n=!0;
    p=!(a==10);
    q=!(a!=b);
    r=(!a ==10);
    printf("%d %d %d %d %d %d %d %d %d\n",
i,j,k,l,m,n,p,q,r);
  return (0);
```

}

పై ప్రోగ్రాంలో ప్రతీ లైనులో ఏమి అవుతుందో తెలుసుకుందాం.

i=!a	**a** వాల్యూ **10**. అంటే ట్రూ. దాని **negation** ఫాల్స్ అంటే సున్నా. అందువల్ల **i** వాల్యూ సున్నా అవుతుంది.
j=!b;	**b** వాల్యూ **-10**. అంటే ట్రూ. దాని **negation** ఫాల్స్. అంటే సున్నా. అందువల్ల **j** వాల్యూ సున్నా అవుతుంది.
k=!c;	**c** వాల్యూ **0**. అంటే ఫాల్స్. దాని **negation** ట్రూ అంటే ఒకటి. అందువల్ల **k** వాల్యూ ఒకటి అవుతుంది.
l=!100;	వాల్యూ **100** అంటే ట్రూ. దాని **negation** ఫాల్స్. అంటే సున్నా. అందువల్ల **l** విలువ సున్నా అవుతుంది.
m=!-100;	వాల్యూ **-100** అంటే ట్రూ. దాని **negation** ఫాల్స్. అంటే సున్నా. అందువల్ల **m** విలువ సున్నా అవుతుంది.
n=!0;	వాల్యూ **0** అంటే ఫాల్స్. దాని **negation** ట్రూ అంటే ఒకటి. అందువలన **n** వాల్యూ ఒకటి అవుతుంది.

p=!(a==10);	a వాల్యూ 10 కాబట్టి, రిలేష్ నల్ ఆపరేటరు వాల్యూ ట్రూ అవుతుంది. దానికి, negation అప్లై చేస్తే ఫాల్స్ వస్తుంది. కాబట్టి p వాల్యూ సున్నా అవుతుంది.
Q=!(a!=b);	a,b వాల్యూలు 10,-10 కాబట్టి, రిలేష్ నల్ ఆపరేటరు వాల్యూ ఫాల్స్ అవుతుంది. దానికి, negation అప్లై చేస్తే ట్రూ వస్తుంది. కాబట్టి q వాల్యూ ఒకటి అవుతుంది.
R=(!a ==10);	a వాల్యూ 10 కాబట్టి, negation అప్లై చేస్తే ఫాల్స్ వస్తుంది. రిలేష్ నల్ ఆపరేటరు వాల్యూ ఫాల్స్ అవుతుంది., కాబట్టి r వాల్యూ సున్నా అవుతుంది.

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా ఉంటుంది.



ఉదాహరణ

ఇంతకు ముందు పాఠాల్లో అయిదుగురిలో ఎంతమంది పాస్ అయ్యారో కనుక్కోవడానికి ప్రోగ్రాం రాశాం. ఇక్కడ, negation వాడి ఎంత మంది ఫెయిల్ అయ్యారో తెలుసుకుందాం.

```
#include<stdio.h>
int main()
{
    int a,b,c,d,e,np;

    printf("Enter five students marks in a test\n");
    scanf("%d%d%d%d%d", &a, &b, &c, &d, &e);
    np=!(a>=35) + !(b>=35) + !(c>=35) + !(d>=35) + !
(e>=35);

    printf("Number of students failed=%d\n", np);
```

```
  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా ఉంటుంది.

```
Enter five students marks in a test
89 22 33 92 31
Number of students failed=3
```

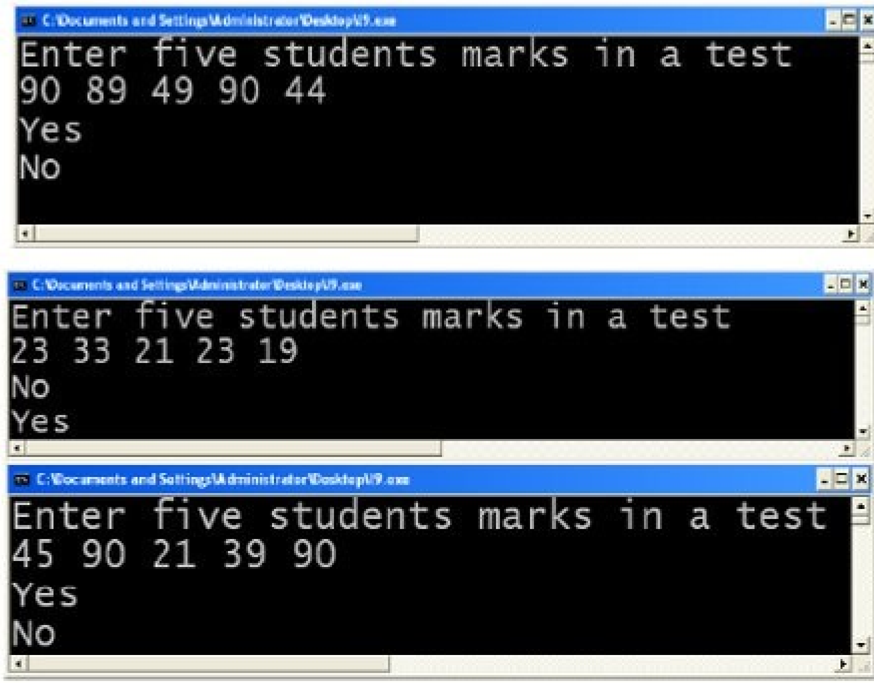
ఉదాహరణ

ఈ కింది ప్రోగ్రాంలో, negation ఆపరేటరు లాజికల్ OR ఆపరేటరులు ఉన్న ఒక expression కు అప్లై చేస్తే ఏమవుతుందో చూపించాం. శాంపిల్ ఇంపుట్, అవుట్ పుట్ లను గమనించండి.

```
#include<stdio.h>
int main()
{
    int a,b,c,d,e,np;

    printf("Enter five students marks in a test\n");
    scanf("%d%d%d%d%d", &a, &b, &c, &d, &e);
    ( ((a>=35) || (b>=35) || (c>=35) || (d>=35) ||
(e>=35)) ) ? printf("Yes\n"):printf("No\n");
    ( !((a>=35) || (b>=35) || (c>=35) || (d>=35) ||
(e>=35)) ) ? printf("Yes\n"):printf("No\n");
  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా ఉంటుంది.



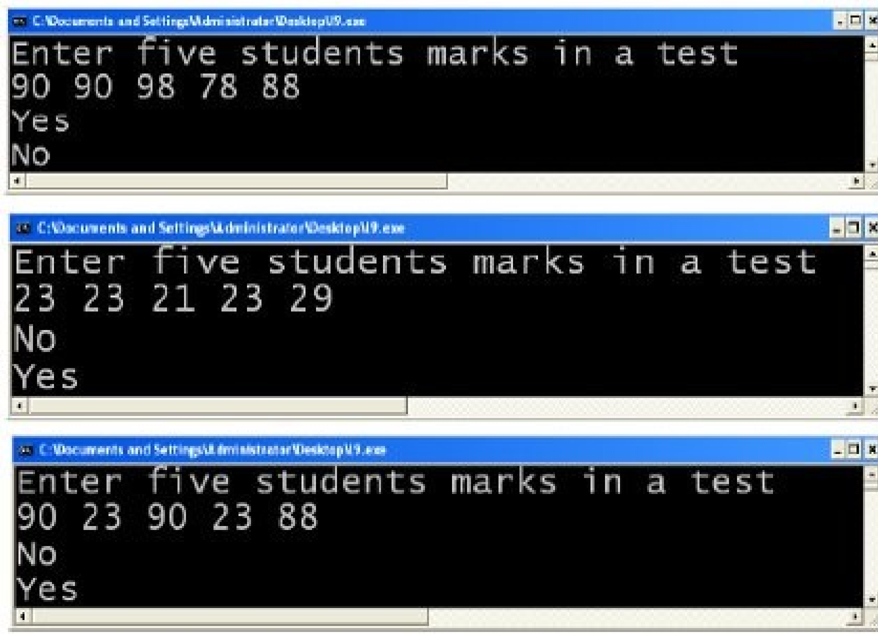
ఉదాహరణ

ఈ కింది ప్రోగ్రాంలో, negation ఆపరేటరు లాజికల్ AND ఆపరేటర్లు ఉన్న ఒక expression కు అప్లై చేస్తే ఏమవుతుందో చూపించాం. శాంపిల్ ఇంపుట్, అవుట్ పుట్ లను గమనించండి.

```
#include<stdio.h>
int main()
{
    int a,b,c,d,e,np;

    printf("Enter five students marks in a test\n");
    scanf("%d%d%d%d%d", &a, &b, &c, &d, &e);
  ( ((a>=35) && (b>=35) && (c>=35) && (d>=35) &&
(e>=35)) ) ? printf("Yes\n"):printf("No\n");
  ( !((a>=35) && (b>=35) && (c>=35) && (d>=35) &&
(e>=35)) ) ? printf("Yes\n"):printf("No\n");
  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా వుంటుంది.



Unary Increment/Decrement Operators

వీటిని integer వైపు వేరియబుల్ కు మాత్రమే వాడవచ్చు. ఇక్కడ కూడా రెండు వైపులు ఉన్నాయి. అవి, postfix increment/decrement, prefix increment/decrement ఆపరేటర్లు. ఉదాహరణకు, ఈ కింది టేబుల్ లో A అనే వేరియబుల్ మీద వీటిని ఎలా వాడవచ్చో చూడండి.

A++	Postfix increment
A--	Postfix decrement
++A	Prefix increment
--A	Prefix decrement

వీటిని ఓ expression లో వాడితే ఏమవుతుందో ఈ కింది ప్రోగ్రాంల ద్వారా తెలుసుకుందాం.

ఉదాహరణ

ఈ ప్రోగ్రాం postfix, prefix increment ఆపరేటర్ల గురించి తెలియజేస్తుంది. ఇక్కడ, b=a++; లో ముందుగా a వాల్యూను b కు ఇచ్చి ఆ తర్వాత a వాల్యూ ఒకటి పెరుగుతుంది. అలాగే, c=++a; లో ముందుగా a వాల్యూ ఒకటి పెరిగి ఆ పెరిగిన వాల్యూను c కు ఇస్తుంది. ఇలాగా postfix, prefix increment ఆపరేటర్లు పని చేస్తాయి.

```
#include<stdio.h>
int main()
{
    int a=10,b,c;

    b=a++;
```

```
    printf("%d %d\n",a,b);
    c=++a;
    printf("%d %d\n", a, c);

  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\9.exe
11 10
12 12
```

ఉదాహరణ

ఈ ప్రోగ్రాం postfix, prefix decrement ఆపరేటర్ల గురించి తెలియజేస్తుంది. ముందుగా a వాల్యూను b కు ఇచ్చి ఆ తర్వాత a వాల్యూ ఒకటి తగ్గుతుంది. అలాగే, c=--a; లో ముందుగా a వాల్యూ ఒకటి తగ్గి ఆ తగ్గిన వాల్యూను c కు ఇస్తుంది. ఇలాగా postfix, prefix decrement ఆపరేటర్లు పని చేస్తాయి.

```
#include<stdio.h>
int main()
{
    int a=10,b,c;

    b=a--;
    printf("%d %d\n",a,b);
    c=--a;
    printf("%d %d\n", a, c);
    scanf("%d",&a);
  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధం ఉటుంది.

```
C:\Documents and Settings\Administrator\Desktop\9.exe
9 10
8 8
```

ఒక expression లో ఒక వేరియబుల్ కు n postfix ఆపరేటర్లు ఉంటే, ఆ వేరియబుల్ ప్రస్తుత వాల్యూతో ఆ expression ను ఎవాల్యూయేట్ చేసి ఆ తర్వాత ఆ వేరియబుల్ వాల్యూ n సార్లు మారుతుంది. అలాగే, ఒక expression లో ఒక వేరియబుల్ కు n prefix ఆపరేటర్లు ఉంటే, ఆ వేరియబుల్ వాల్యూ n సార్లు

ముందుగా మారుతుంది, ఆ మారిన వాల్యూతో expression ఎవాల్యుయేట్ అవుతుంది.

ఉదాహరణ

పైన వివరించిన అంశాలను ఈ కింది ప్రోగ్రాం ద్వారా తెలుసుకుందాం.
ఇక్కడ, b=a++ + a++ + a++;
తో a కు మూడు postfix increment లు వాడాం. కాబట్టి, ప్రస్తుత a వాల్యూ 10 తో a+a+a, అంటే 30 ను b కు ఇచ్చి ఆ తర్వాత a వాల్యూ మూడు సార్లు పెరుగుతుంది. అంటే 13 అవుతుంది. అలాగే, c=++a + ++a + ++a;
లో a కు మూడు prefix increment లు ఉన్నాయి. కాబట్టి a మూడు సార్లు పెరుగుతుంది. అంటే, 16 అవుతుంది. అప్పుడు, a+a+a, అంటే 16+16+16=48 కాలిక్యులేట్ చేసి c కు ఇస్తుంది. అలాగే, c=++a + ++a + ++a;
లో a కు ఒక prefix increment, postfix increment లు ఉన్నాయి. కాబట్టి, a ముందు పెరుగుతుంది. అంటే, 17 అవుతుంది. దానితో a+a అంటే, 34 కాలిక్యులేట్ చేసి d కి ఇస్తుంది. ఆ తర్వాత, a వాల్యూ ఒకటి తగ్గి 16 అవుతుంది.

```
#include<stdio.h>
int main()
{
    int a=10,b,c,d;

    b=a++ + a++ + a++;
    printf("%d %d\n",a,b);
    c=++a + ++a + ++a;
    printf("%d %d\n", a, c);
    d=++a + a--;
    printf("%d %d\n",a,d);

  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\19.exe
13 30
16 46
16 34
```

ఉదాహరణ

ఈ ప్రోగ్రాం లో postfix, prefix ఆపరేటర్లను మామూలుగా, అంటే ఏ expression లోనూ కాకుండా వాడాం. వీటినే standalone statements అని

అంటారు. అప్పుడు postfix, prefix కు మధ్య ఏ వ్యత్యాసం ఉండదు. అంటే a++ లేక ++a అనేవి రన్ అయిన తర్వాత a వాల్యూ ఒకటిగా ఉంటుంది. దీనిని ఈ కింద వివరించిన ప్రోగ్రాం ద్వారా తెలుసుకుందాం.

```
#include<stdio.h>
int main()
{
    int a=10,b=10;

    a++;
    ++b;

    printf("%d %d\n",a,b);
  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా ఉంటుంది.



ఉదాహరణ

ఈ కింది ప్రోగ్రాం 0, 0 ను రిజల్ట్ లాగా ఇస్తుంది. ఎందువల్ల?

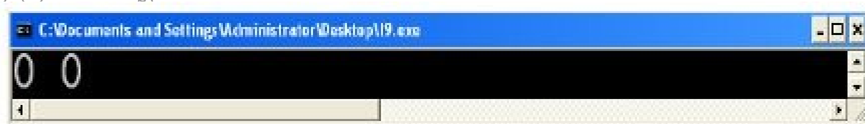
```
#include<stdio.h>
int main()
{
    int a=0,b=0,c,d;

    c=(a++&&b++);
    d=(b&&++c);


    printf("%d %d\n",c,d);

  return (0);
}
```

పై ప్రోగ్రాం రిజల్ట్ ఈ విధంగా ఉంటుంది.



ముందుగా లాజికల్ ఆపరేటర్ && గురించి గుర్తుకు తెచ్చుకోవాలి. దీనిని ఎవాల్యుయేట్ చేస్తున్నప్పుడు మొదటి ఆపరాండ్ ఫాల్స్ అయితే రెండోది ఎవాల్యుయేట్ అవదు. ఎందుకంటే అది ట్రూ అయినా ఫాల్స్ అయినా రిజల్ట్ ఫాల్స్ అవుతుంది. అందువల్ల, ఈ ప్రోగ్రాంలో c=(a++&&b++); ఎవాల్యుయేట్ చేస్తున్నప్పుడు a సున్నా అయినందువల్ల c వాల్యూ సున్నా అవుతుంది. b++ రన్ అవదు. కాబట్టి b వాల్యూ సున్నాగానే ఉంటుంది. అలాగే, d=(b&&++c); ఎవాల్యుయేట్ చేస్తున్నప్పుడు b సున్నా కాబట్టి, లాజికల్ ఆపరేటరు రిజల్ట్ సున్నా అవుతుంది. కాబట్టి d కూడా సున్నా అవుతుంది.

Character Variables అంటే ఏమిటి?

ముంద chapter లలో ఇప్పటివరకు మనం calculations లలో int, float, long, double అనే variables ను ఉపయోగించడం తెలుసుకున్నాం. ఒక్కోసారి మనం మనుషుల పేర్లను, చిరునామాలను, బ్లడ్ గ్రూపులను, student ల గ్రేడ్ లను, మనుషుల DNA లను ప్రోసెస్ చేయాల్సి వస్తుంది. అప్పుడు character variables అనే వాటిని వాడాల్సి ఉంటుంది. ఒక integer variable లోకి 12 అయినా 33321 అయినా, మరేదయినా integer నంబరును store చేయవచ్చు. అలాగే, ఒక float variable లోకి 1.2223 అయినా 322.122 అయినా, మరేదయినా float నంబరును store చేయవచ్చు. అలాగే, ఒక character variable లోకి ఈ కింద ఇచ్చిన ASCII table 1 లోని ఒక symbol ను మాత్రమే స్టోర్ చేయవచ్చు. ఈ కింద ఇచ్చిన table 1. లో, C లాంగ్వేజీలో వాడగలిగిన symbols వాటి decimal, Hex, Octal values ఉన్నాయి. వీటి గురించి మరింతగా తర్వాత తెలుసుకుందాం. C లాంగ్వేజీలో ఒక character variable కు ఒక byte allocate అవుతుంది.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Htm	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Table 1: ASCII Table

ఒక integer variable కు ఒక integer value(constant)ను ఇస్తే ఆ integer value(constant) కి సంబంధించిన binary code ఆ variable కు allocate చేసిన memory లో స్టోర్ అవుతుంది. ఉదాహరణకు, int x=109 లో 109 కి సంబంధిచిన binary code(01101101), x కు allocate చేసిన memory లో స్టోర్ అవుతుంది. ఇది float variables కు కూడా వర్తిస్తుంది. అలాగే, ఒక character variable కు ఒక symbol assign చేస్తే, దాని ASCII code (unique code) ఆ variable కు allocate చేసిన memory లో స్టోర్ అవుతుంది. దీనిని మనం మార్చలేం. ఏ రెండు symbols కూ ఒకే ASCII code ఉండదు.

మామూలుగా, 1278 లాంటి వాటిని integer constants అని, 12.333 లాంటి వాటిని float constants అని అంటారు. అలాగే, ఏ symbol అయినా single quotes మధ్యలో ఉంటే (ఉదాహరణకు 'A','X', '9', '\n'), దానిని character constant అని అంటారు. ఉదాహరణకు, ఈ statement ను గమనించండి.

char v='A';

ఇక్కడ A అనే symbol(character constant)ను, v అనే character variable కు assign చేస్తున్నాం. ఇప్పుడు, v value ఏమిటి అంటే symbol A లేక symbol capital A అని చెబుతాం. అప్పుడు ఆ symbol ASCII code, v అనే variable memory లో స్టోర్ అవుతుంది. అంటే, 01000001(అంటే డెసిమల్ 65) అనే binary నంబరు v variable memory లో స్టోర్ అవుతుంది.

దీనినే, symbol A ASCII code 65 అని కూడా చెప్పవచ్చు. అలాగే, capital letters (A-Z) ASCII codes: 65-90, lower case letters ASCII codes: 97-122, digits 0-9 ASCII codes 48-57. అన్ని కంప్యూటర్లూ ఇలాగే తీసుకుంటాయి. కాబట్టి ఏ కంప్యూటరులో type చేసినా, మనం ఏ కంప్యూటరులో చూసినా ఒకే రకంగా ఉంటాయి. దీనినే ASCII compatibility అని అంటారు. ఇంతకు ముందే ఒక character variable కు ఒక symbol ఇవ్వవచ్చు అని తెలుసుకున్నాం. ఇది 0-9 కు కూడా వర్తిస్తుంది. ఇవి కూడా అక్షరాలే కదా? అంటే, char v='2'; అనే దానిలో digit 2 ను v అనే variable కు ఇస్తున్నామని అర్థం. అప్పుడు, 2 ASCII code (00110010), v variable memory లో స్టోర్ అవుతుంది. అంటే symbol 2 ASCII code 00110010. ఇక్కడ మనకు కొద్దిగా confusion వస్తుంది. Integer 2, digit 2 లేక symbol 2 కు మధ్య వ్యత్యాసం ఏమిటి అని?. కంప్యూటరులో వీటికి వ్యత్యాసం ఉంది. కింద ఇచ్చిన table 2 ఈ వ్యత్యాసాన్ని విశదీకరిస్తుం ది. Integer ఇస్తే కంప్యూటరులో ఎలా స్టోర్ అవుతుంది, digit ఇస్తే ఏం స్టోర్ అవుతుందో యీ table 2 ద్వారా తెలుసుకోవచ్చు.

Integer	Binary Code	Digit	Binary Code (Decimal values)
0	00000000	'0'	00110000 (48)
1	00000001	'1'	00110001 (49)
2	00000010	'2'	00110010 (50)
3	00000011	'3'	00110011 (51)
4	00000100	'4'	00110100 (52)
5	00000101	'5'	00110101 (53)
6	00000110	'6'	00110110 (54)
7	00000111	'7'	00110111 (55)
8	00001000	'8'	00111000 (56)
9	00001001	'9'	00111001 (57)

Table 2 Difference between integers 0-9 and character 0-9 or digit 0-9.

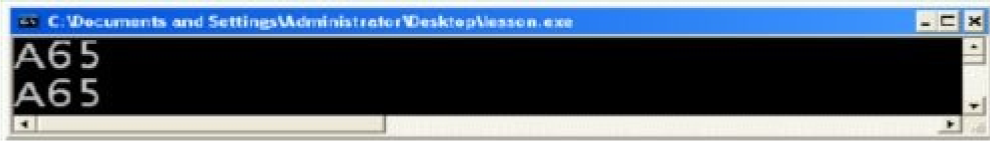
C ప్రోగ్రాంలోని arithmetic expressions లో, character variables, character constants ను వాడితే, వాటి ASCII codes ను ఉపయోగించి expressions ను evaluate చేస్తుంది. ఒక విధంగా character variables, character constants ను integers తో సమాంగా అనుకుంటాం. కొన్ని సార్లు దాని స్థానం లో దీన్ని, దీని స్థానం లో దాన్ని ఉపయోగించడం సహజం. ఈ కింద ఇచ్చిన ప్రోగ్రాం దీన్ని explain చేస్తుంది.

Example 1: ఈ ప్రోగ్రాంలో రెండు variables ను వాడాం, అందులో ఒకటి character, ఇంకొకటి integer type. వాటికి కొన్ని values ను assign చేసి వాటిని %c, %d format లను వాడి print చేయించాం. అంటే, ఈ ప్రోగ్రాం

character స్థానం లో integer ను , integer స్థానంలో character ను వాడవచ్చు అని తెలుసుకోవచ్చు.

```
#include<stdio.h>
int main()
{
    char v='A';
    int p=65;
    printf("%c%d\n", v,v);
    printf("%c%d\n", p,p);
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఇలా ఉంటుంది.



Example 2: Capital letter ను input గా తీసుకొని దానికి సమానమైన lower case letter ను ఇవ్వడానికి ప్రోగ్రాం.

Solution: ASCII code table నుంచి మనం ఒక విషయాన్ని గ్రహించవచ్చు. అదేమిటి అంటే, capital అక్షరాల, lower case అక్షరాల ASCII code ల మధ్య భేదం 32. కాబట్టి తీసుకున్న capital అక్షరానికి 32 కలిపితే దానికి సమానమైన lower case అక్షరం వస్తుంది. అలాగే lower case అక్షరంలో నుంచి 32 తీసేస్తే దానికి సమానమైన upper case అక్షరం వస్తుంది. ఈ కింద ఇచ్చిన ప్రోగ్రాం దీన్ని వఉపయోగించుకొని పని చేస్తుంది.

```
#include<stdio.h>
int main()
{
    char v, p;
    printf("Enter an upper case character\n");
    scanf("%c", &v);
    p = v+32;
    printf("%c%c\n", v, p);
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఇలా ఉంటుంది.



Example 3: ఒక digit ను ఒక character variable లోకి రీడ్ చేసి దాని విలువను 17 తో multiply చేసి వచ్చిన వాల్యూను ప్రింట్ చేయడానికి ప్రోగ్రాం.

Solution: Table 2 ను చూస్తే మనకు ఒక విషయం తెలుస్తుంది. Digits ASCII code విలువలు 48 నుంచి 57 వరకు ఉన్నాయి. ఈ విలువల నుంచి 48 తీసేస్తే, మనకు 0-9 వస్తాయి. అంటే, ఒక digit ASCII code లో నుంచి digit value కావాలంటే 48 తీసేయాలి. ఈ కింద ఇచ్చిన ప్రోగ్రాం దీన్ని ఉపయోగించుకొని పని చేస్తుంది. ఇందులో 48 తీయకుండా చేస్తే ఏమవుతుందో కూడా ఒక variable (p) ద్వారా చూపించాం. మనకు రావాల్సిన వాల్యూ q లో ఉంటుంది.

```
#include<stdio.h>
int main()
{
    char v; int p, q;
    printf(“Enter a digit (0-9)\n”);
    scanf(“%c”, &v);
    p = 17*v;
    q = 17*(v-48);
    printf(“%c %d %d\n”, v, p, q);
    return (0);
}
```

పై ప్రోగ్రాం రన్ చేసినప్పుడు స్క్రీన్ ఈ విధంగా ఉంటుంది.



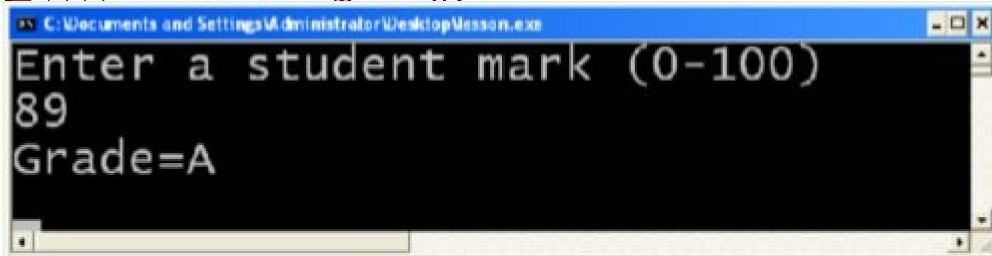
Example 4: ఒక విద్యార్థి test marks (0-100) రీడ్ చేసి అతడి గ్రేడ్ ను print చేయడానికి ప్రోగ్రాం ఇది. ఎన్ని మార్కులు వస్తే ఏ గ్రేడ్ రావాలో ఈ కింది table చూపిస్తుంది.

Marks	Grade
>= 80	A
60-80	B
40-60	C
20-40	D
<20	E

Solution: ఇచ్చిన టేబుల్ ను చూస్తే మనకు ఒక విషయం తెలుస్తుంది. అదేమిటంటే, ప్రతి 20 మార్కులకు ఒక గ్రేడ్ మారుతుంది. కాబట్టి మనం ఇచ్చిన marks లో ఎన్ని 20 లు ఉన్నాయో కనుక్కోని దానిని నాలుగు నుంచి తీసివేసి, symbol A కు కలిపితే మనకు కావాల్సిన గ్రేడ్ వస్తుంది. కాని ఈ logic 100 ఇస్తే పని చేయదు. 100 తో వేరే symbol answer లాగా వస్తుంది. దీనిని కరెక్ట్ చేయడానికి, marks 100 అయితే 1 లేకపోతే 0 వచ్చే విధంగా (m==100) అనే ఒక చిన్న logic అదనంగా ఈ కింది ప్రోగ్రాంలో ఉపయోగించాం.

```
#include<stdio.h>
int main()
{
    int m;
    char v;
    printf("Enter a student mark (0-100)\n");
    scanf("%d", &m);
    v = 'A'+ (4- m/20 )+( m==100 );
    printf("Grade=%c\n", v);
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఇలా ఉంటుంది.



అదే ప్రోగ్రాంను ఈ విధంగా కూడా రాయవచ్చు. ఇక్కడ, ముందు ప్రోగ్రాంలో వాడిన v = 'A'+ (4- m/20)+(m==100)లో A బదులు 65 పెట్టి 4 ను బయటకు లాగి simplify చేస్తే వచ్చింది.

```
#include<stdio.h>
int main()
{
    int m;
    char v;
    printf("Enter a student mark (0-100)\n");
    scanf("%d", &m);
    v = 69-m/20+(m==100);
    printf("Grade=%c\n", v);
    return (0);
}
```

Example 5: ఒక విద్యార్థి test గ్రేడ్ రీడ్ చేసి అతడి గ్రేడ్ పాయింట్ ను print చేయడానికి ప్రోగ్రాం ఇది. ఏ గ్రేడ్ పాయింట్ వస్తే ఏ గ్రేడ్ ఇవ్వాలో ఈ కింది table చూపిస్తుంది.

Grade	Points
A	10
B	8
C	6
D	4
E	2

Solution: ఇచ్చిన టేబుల్ ను చూస్తే మనకు ఒక విషయం తెలుస్తుంది. అదేమిటంటే, ప్రతి గ్రేడ్ change కు 2 గ్రేడ్ పాయింట్ లు వస్తున్నాయి. మనం A గ్రేడ్ (ASCII code, 65) ఇస్తే మనకు 10 రావాలి, మనం B (ASCII code, 66) గ్రేడ్ ఇస్తే మనకు 8 రావాలి. గమనిస్తే, మనం ఇచ్చిన గ్రేడ్ ను 70 లో నుంచి తీసివేసి, 2 తో హెచ్చవేస్తే, మనకు కావాల్సిన గ్రేడ్ పాయింట్ లు వస్తాయి. ఈ కింద ఇచ్చిన ప్రోగ్రాం దీన్ని ఉపయోగించి పని చేస్తుంది.

```
#include<stdio.h>
int main()
{
    char v;
    int p ;
    printf("Enter a student grade in a test (A-E)\n");
    scanf("%c", &v);
    p = 2*(70-v);
    printf("Points=%d\n", p);
```

```
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఇలా ఉంటుంది.



Example 6: ఒక విద్యార్థి అయిదు టెస్టుల్లో గ్రేడ్ లను తీసుకొని అతడి గ్రేడ్ పాయింట్ యావరేజ్ ను ప్రింట్ చేయాలి. గ్రేడ్ పాయింట్ రూల్స్ ప్రీవియస్ Example లో ఇచ్చినవి తీసుకోవాలి. ఉదాహరణకు AABCA input లాగా ఇస్తే, రావాల్సిన యావరేజ్ (10+10+8+6+10)/5=8.8. అలాగే, ACCBD input లాగా ఇస్తే, యావరేజ్ (10+6+6+8+4)/5=34/5=7.8 రావాలి.

Solution: ఇం తకు ముందు ప్రోగ్రాంలో ఒక గ్రేడ్ తీసుకొని గ్రేడ్ పాయింట్ ప్రింట్ చేశాం. ఈ ప్రోగ్రాంలో అయిదు గ్రేడ్ లు తీసుకొని వాటి యావరేజ్ కావాలి కాబట్టి ఇంతకు ముందు ప్రోగ్రాంలో వాడిన logic ను ప్రతి గ్రేడ్ మీద వాడి మొత్తం లెక్కగట్టి యావరేజ్ కనుక్కోవాలి. ఈ కింద ఇచ్చిన ప్రోగ్రాంలో ప్రతి టెస్ట్ లో గ్రేడ్ ను substitute చేసి simplify చేస్తే వచ్చిన expression ను వాడాం. ఉదాహరణకు AABCA input గా ఇస్తే, P='A', Q='A', R='B', S='C', T='A' అవుతాయి. అప్పుడు GPA = 2*(70-(P+Q+R+S+T)/5.0); expression GPA=2*(70-(65+65+66+67+65)/5.0)=6.8 అవుతుంది.

```
#include<stdio.h>
int main()
{
    char P, Q, R, S, T;
    float GPA;
    printf("Enter a student grades in 5 tests\n");
    scanf("%c%c%c%c%c", &P, &Q, &R, &S, &T);
    GPA = 2*(70-(P+Q+R+S+T)/5.0);
    printf("Grade Point Average=%f\n", GPA);
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson.exe
Enter a student grades in 5 tests
AABCA
Grade Point Average=8.800000
```

```
C:\Documents and Settings\Administrator\Desktop\lesson.exe
Enter a student grades in 5 tests
ACCBD
Grade Point Average=6.800000
```

Example 7: ఒక విద్యార్థి కి సంబంధించి అయిదు టెస్టుల్లో గ్రేడ్ లను తీసుకొని అతడు పాస్ అయ్యాడా లేదా ఫెయిల్ అయ్యాడా చెప్పాలి. పాస్ అవ్వాలంటే గ్రేడ్ పాయింట్ యావరేజ్ 5 ఉండాలి. అంతే కాకుండా ప్రతి టెస్టులో at least D గ్రేడ్ రావాలి. గ్రేడ్ పాయింట్ రూల్స్ ఇంతకు ముందు ఇచ్చిన ప్రోగ్రాం ప్రకారం తీసుకోవాలి.

Solution: ఇంతకు ముందు ప్రోగ్రాం చూస్తే ఇది చిన్న మార్పుతో ఉన్న ప్రోగ్రాం. ఇక్కడ కూడా గ్రేడ్ పాయింట్ యావరేజ్ కావాలి. ప్రతి టెస్టులో D గ్రేడ్ రావాలి అనే దానిని మనం ఇలా కూడా చెప్పవచ్చు కదా? ఏ టెస్టులో కూడా E గ్రేడ్ రాకూడదని?. కాబట్టి ఈ ప్రోగ్రాంలో ఎన్ని టెస్టుల్లో E గ్రేడ్ వచ్చిందో కనుక్కొని ఒక variable(np)లో స్టోర్ చేస్తాం. దీనిని, గ్రేడ్ పాయింట్ యావరేజ్ ను ఉపయోగించి పాస్ లేదా ఫెయిల్ చెబుతాం.

```
#include<stdio.h>
int main()
{
    char p, q, r, s, t;
    float gpa; int np;
    printf("Enter a student graded in 5 test\n");
    scanf("%c%c%c%c%c", &p, &q, &r, &s, &t);
    gpa = 2*(5-((p+q+r+s+t)/5.0-'A'));
    np=(p=='E')+(q=='E')+(r=='E')+(s=='E')+ (t=='E');
    /* This gives in how many of the tests candidate has
got 'E' grades.*/
    ((np==0)*(gpa>=5))?printf("Passed\n"):
printf("Failed \n");
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఈ విధంగా ఉంటుంది.

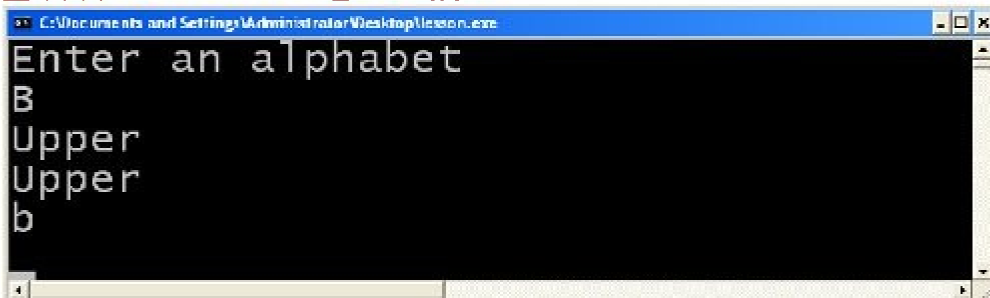
```
C:\Documents and Settings\Administrator\Desktop\lesson.exe
Enter a student  grades in 5 test
AABCA
Passed
```

Example 8: Alphabet ను రీడ్ చేసి అది upper case లేక lower case అని ప్రింట్ చేయాలి. దానికి సమానమైన వేరే case Alphabet ను కూడా ప్రింట్ చేయాలి.

Solution: దీనిని రెండు విధాలుగా చేయవచ్చు. ఇచ్చిన Alphabet, A-Z మధ్యలో ఉందా లేదా అని చెక్ చేసి ఉండి ఉంటే upper అని లేకపొతే lower అని చెప్పవచ్చు. ఇంకోవిధంగా చెప్పాలంటే, ఇచ్చిన Alphabet, a-z మధ్యలో ఉందా అని చెక్ చేసి, ఉంటే lower అని లేకపొతే upper అని చెప్పడం. ఈ కింది ప్రోగ్రాంలో రెండూ ఉపయోగించాం. అలాగే, ఇంతకు ముందు చేసినట్లే, lower ను upper లోకి చేయడానికి 32 తీసేస్తాం, అలాగే upper ను lower లోకి చేయడానికి 32 కలుపుతాం.

```
#include<stdio.h>
int main()
{
    char v;
    printf("Enter an alphabet\n");
    scanf("%c", &v);
    ((v>='A')*(v<='Z'))?
printf("Upper\n"):printf("Lower\n");
    ((v>='a')*(v<='z'))?
printf("Lower\n"):printf("Upper\n");
    v=((v>='A')*(v<='Z'))? v+32 : v-32;
    printf("%c\n", v);
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఈ విధంగా ఉంటుంది.



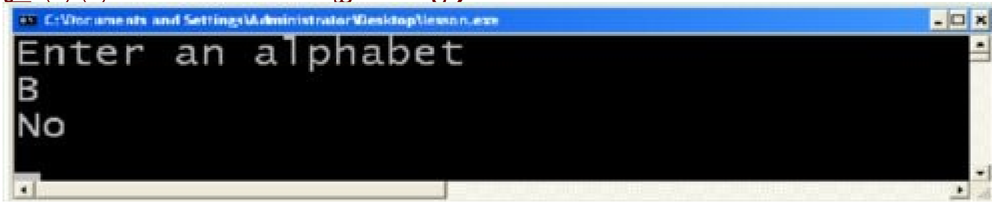
Example 9: ఒక character ను రీడ్ చేసి vowel అయితే "YES" అని లేకపోతే "NO" అని ప్రింట్ చేయాలి.

Solution: ముందుగా ఇచ్చిన character లోయర్ అయితే దానిని అప్పర్ లోకి మార్చి, అ తర్వాత vowels (AEIOU లతో) పోలుస్తాం.

```
#include<stdio.h>
int main()
```

```
{
    char v;
    printf("Enter an alphabet\n");
    scanf("%c",&v);
    v = ((v>='a')*(v<='z'))?v-32:v;
    ((v=='A')+(v=='E')+(v=='I')+(v=='O')+(v=='U'))?
printf("Yes\n"):printf("No\n");
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఈ విధంగా ఉంటుంది.



ESCAPE Characters (OR) Special Characters

C లాంగ్వేజీలో ఈ కింద ఇచ్చిన వాటిని కూడా characters లాగా తీసుకుంటుంది. అంతే కాకుండా keyboard మీద ఉన్న ప్రతి అక్షరానికి ఒక character ఉంటుంది. కొన్ని మనం చూడలేం, కానీ వాటి effect ను చూడవచ్చు. ఉదాహరణకు, Left Arrow, Right Arrow, Page Up, Page Down, అన్నీ కూడా system పాయింట్ లో ఒక character. వీటినన్నింటినీ ESCAPE Characters అని అంటారు.

```
\n →new line;
\a →beep;
\f →form feed;
\t →tab;
\b →backspace;
```

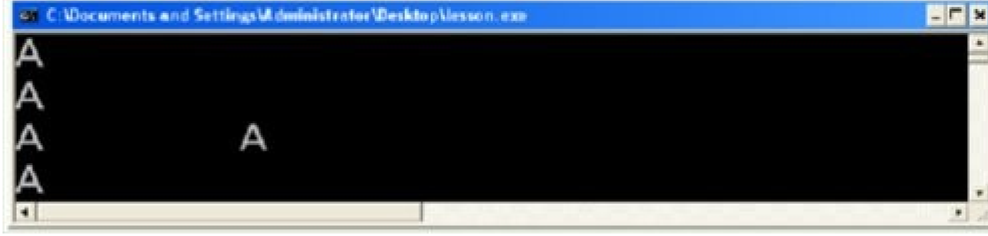
Example 10: ఈ ప్రోగ్రాం "Escape Characters" ఎలా ఉపయోగించాలో తెలియజేస్తుంది. ఈ ప్రోగ్రాం output కింద ఇచ్చినట్లు ఉంటుంది. అంతే కాకుండా బీప్ sound లు మూడు సార్లు మొదటి printf తో వస్తాయి.

```
A
A
A A
A
```

```
#include<stdio.h>
int main()
{
    char p='A', q='\n', r='\t', s='\b', t='\a';
```

```
    printf("%c%c%c", t, t, t);
    printf("%c%c%c\n", p, q, p);
    printf("%c%c%c\n", p, r, p);
    printf("%c%c%c\n", p, s, p);
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఈ విధంగా ఉంటుంది.



Using Ready made functions

Character లను ప్రోసెస్ చేయడానికి readymade functions కూడా ఉన్నాయి. ఉదాహరణకు, “isupper” అనే function ఒక character ను తీసుకొని అది upper case character అయితే 1 return చేస్తుంది, లేకపోతే 0 return చేస్తుంది. ఒక విషయం గుర్తుంచుకోవాలి మనం. function లోకి ఒక character పంపాలి అంటే అది character variable అయినా కావచ్చు, character constant అయినా కావచ్చు, character expression అయినా కావచ్చు. ఈ function లను వాడాలంటే “Turbo C” లో <ctype.h> header ను మొదటలో include చేయాలి. అదే, Unix/Linux లో ఉన్న gcc compiler లో లేదా Dev C++/Codeblocks compiler లలో అయితే <stdlib.h> header ను మొదటలో include చేయాలి. కింద ఇచ్చిన టేబుల్ లో వాటిని ఎలా వాడాలో, అవి ఏం చేస్తాయో వివరంగా ఉంది.

Function పేరు	Function ఏం చేస్తుంది?	character v='X'; అనుకొని, ఎలా Function ను కాల్ చేయ్యాలి?
isuuper	మనం పంపిన character A-Zలలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	isupper(v) or isupper ('P')
islower	మనం పంపిన character a-z లలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	islower(v) or islower ('P')
isalpha	మనం పంపిన character A-Z,a-z లలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	isalpha(v) or isalpha ('t')
isdigit	మనం పంపిన character 0-9 లలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	isdigit(v) or isdigit('8')
isalnum	మనం పంపిన character A-Z,a-z,0-9 లలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	isalnum(v) or isalnum ('x')
isspace	మనం పంపిన character, Space, \t (TAB), \n(newline) లలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	isspace(v) or isspace ('\t')

ispunct	మనం పంపిన character, punctuation character లలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	ispunct(v) or ispunct ('$')
isxdigit	మనం పంపిన character A-F0-9 లలో ఏదైనా అయితే ట్రూ రిటర్న్ చేస్తుంది, లేకపోతే ఫాల్స్ రిటర్న్ చేస్తుంది.	isxdigit(v) or isxdigit ('D')
toupper	మనం పంపిన character, lower అయితే దానికి సమానమైన upper case character వస్తుంది, లేకపోతే పంపిన character అలాగే రిటర్న్ అవుతుంది.	toupper(v) or toupper ('p')
tolower	మనం పంపిన character, upper అయితే దానికి సమానమైన lower case character వస్తుంది, లేకపోతే పంపిన character అలాగే రిటర్న్ అవుతుంది	tolower(v) or tolower ('j')

Table 3: Character manipulation functions

Example 11: ఈ కింది ప్రోగ్రాం ఒక Character ను ఇన్ పుట్ లాగా తీసుకొని అది అప్పరా లేదా లోయరా అని ప్రింట్ చేసి దానిని వేరే case లోకి మార్చి ప్రింట్ చేస్తుంది. ఇదే ప్రోగ్రాం ఇంతకు ముందు కూడా రాశాం. కానీ ఇక్కడ readymade functions ను వాడి చేస్తున్నాం.

```
#include<stdio.h>
#include<ctype.h>
    int main()
{
    char v;
    printf("Enter character\n");
    scanf("%c",&v);
    (isupper(v))?printf("Upper\n"):printf("Not Upper\n");
    v=isupper(v)?tolower(v):toupper(v);
    printf("%c\n", v);
    return (0);
}
```

పై ప్రోగ్రాంను రన్ చేసినప్పుడు స్క్రీన్ ఈ విధంగా ఉంటుంది.



ఈ అధ్యాయం లో character variables అంటే ఏమిటి, వాటిని ఎలా వాడాలి అని తెలుసుకున్నాం. అలాగే వాటితో ఉపయోగించగలిగిన readymade functions గురించి నేర్చుకున్నాం.

దిశను మార్చే కంట్రోల్ స్ట్రక్చర్స్

ఇ ం తకు ముం దు పాఠాల్లో వివిధ రకాలయిన variables ను వాడటం, వాటితో operators ను వాడటం తెలుసుకొన్నాము. రాసిన ప్రోగ్రాములు అన్ని ం టిలోను ఒక statement అయిన తర్వాత next statement రన్ అయ్యాయి. అంటే ప్రోగ్రాం sequential లేదా serial గా రన్ అయినట్లు. అలా కాకుం డా, ప్రోగ్రాం control ఒక statement అయిన తర్వాత కొన్ని statements ను వదలి వేరొక statement కు వెళ్ళాల ం టే, control structures ను వాడుతాం. అంటే ప్రోగ్రాం execution దిశను మార్చడానికి control structures ను వాడుతాం. C లాం గ్వేజీలో ఉపయోగించే control structures కింద ఉన్నాయి.. వాటిని ఎలా వాడాలో ఒక్కోదాని గురిం చి వివరంగా తెలుసుకొం దాం.

1. if condition,
2. goto statement,
3. switch condition,
4. loops, etc.,.

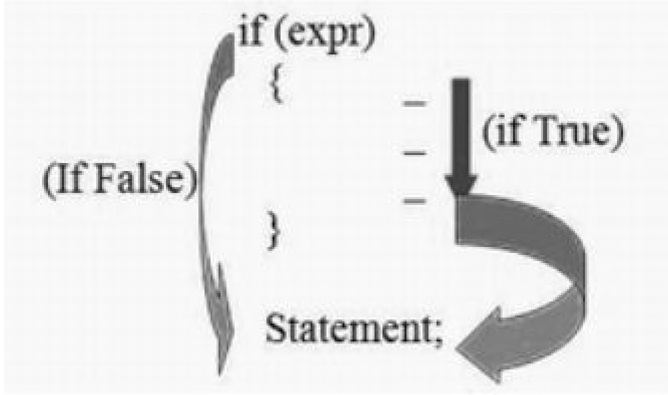
1 The if condition

ఈ if condition అనేది చాలా సులభమైనది. C లాం గ్వేజీ లో ఎక్కువగా ఉపయోగించే control structure. దీనిని మూడు విధాలుగా ఉపయోగించవచ్చు.

1'st Style

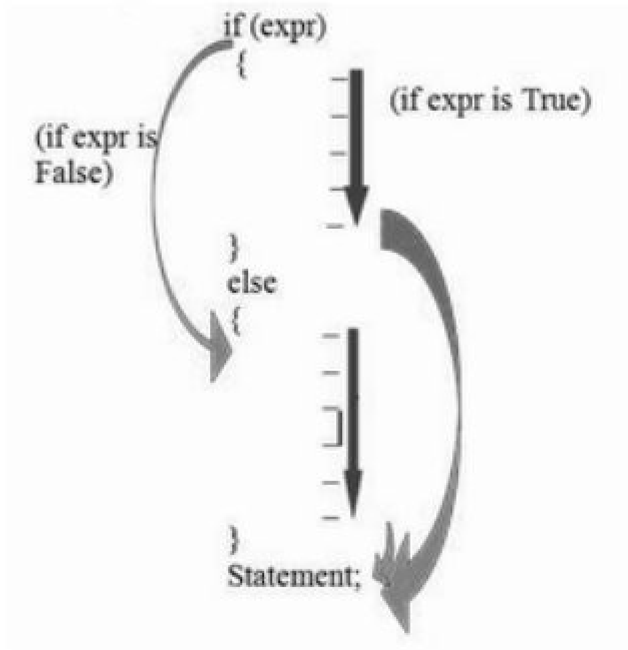
ఈ పద్ధతిలో if లో expr కనుక true అయితే '{' మరియు '}' మధ్యలో ఉం డే statements రన్ అయిన తర్వాత ప్రోగ్రాం control Statement కు వెళ్తుం ది. అదే expr false అయితే program control నేరుగా Statement వెళ్తుం ది. ఇక్కడ '{' మరియూ '}' మధ్యలో వుం డే statements ను if బ్లాక్ అని

అ০ టారు.

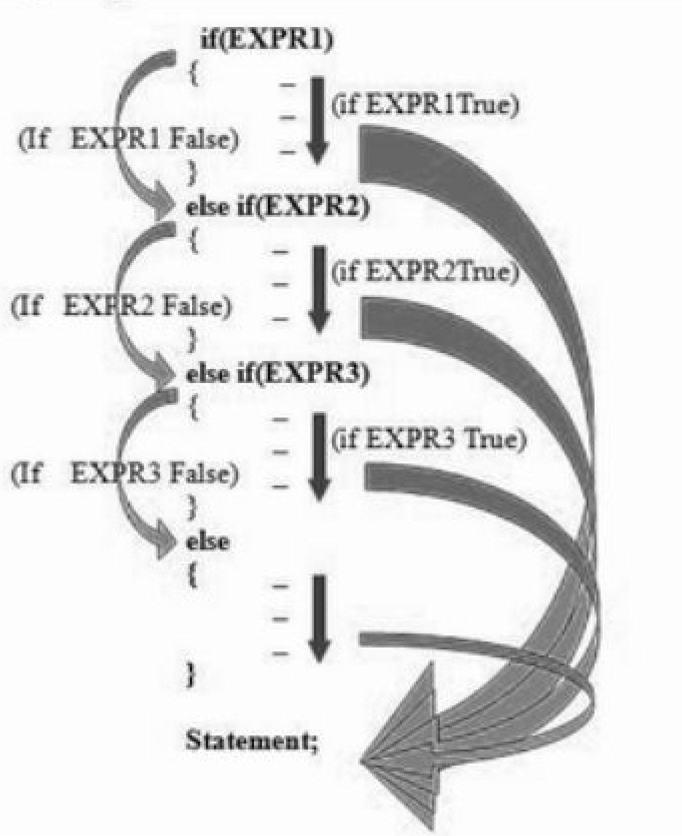


2'nd Style

ఈ పద్ధతిలో if లో expr కనుక true అయితే మొదటి '{' మరియు '}' మధ్యలో ఉ০డే statements రన్ అయిన తర్వాత ప్రోగ్రాం control Statement కు వెళ్తు০ది. అదే expr false అయితే, రె০డవ '{' మరియు '}' మధ్యలో ఉ০డే statements రన్ అయిన తర్వాత program control నేరుగా Statement కు వెళ్తు০ది. ఇక్కడ మొదటి '{' మరియు '}' మధ్యలో వు০డే statements ను if బ్లాక్ అని, రె০డవ '{' మరియూ '}' మధ్యలో వు০డే statements ను else బ్లాక్ అని అ০టారు.



3rd Style



ఈ పద్ధతిలో if లో EXPR1 కనుక true అయితే మొదటి '{' మరియు '}' మధ్యలో ఉండే statements రన్ అయిన తర్వాత ప్రోగ్రాం control Statement కు వెళ్తుంది, అదే EXPR1 false అయితే, else if (EXPR2) కు ప్రోగ్రాం కంట్రోల్ వెళ్తుంది. EXPR2 కనుక true అయితే దాని తర్వాత ఉన్న statement ల బ్లాక్ ను రన్ చేసి Statement కు వెళ్తుంది, లేకుంటే else if (EXPR3) కు ప్రోగ్రాము కంట్రోల్ వెళ్తుంది,. EXPR3 కనుక true అయితే దాని తర్వాత ఉన్న statements బ్లాక్ ను రన్ చేసి Statement కు వెళ్తుంది, లేకుంటే else కు ప్రోగ్రాము కంట్రోల్ వెళ్ళి ఆ తర్వాత Statement కు వెళ్తుంది . దీనిని nested if statement అని అంటారు. మనం else if statement లను ఎన్నైనా వాడుకోవచ్చు. అలాగే చివరలో ఎప్పుడూ else బ్లాక్ వుండాల్సిన అవసరం కూడా లేదు.

ఏ బ్లాక్({, } మద్యలో వుండేది) లో అయినా ఒక్క statement మాత్రమే ఉంటే, బ్రాకెట్స్ తీసివేయవచ్చు (ఈ statement main బ్లాక్ కు, function బ్లాక్ లకు వర్తించదు).

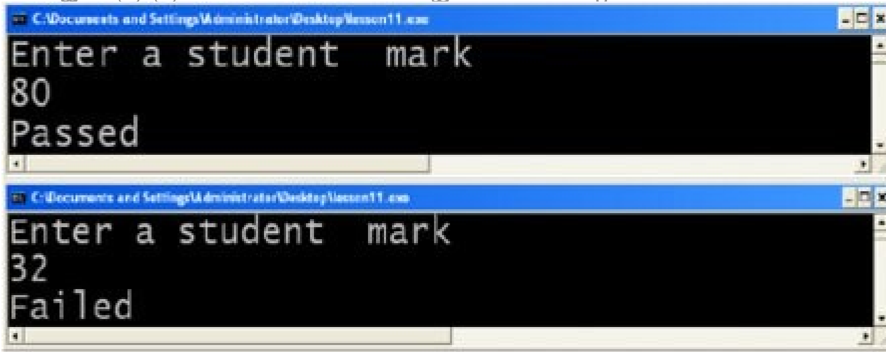
Example 1: ఒక student కు వచ్చిన test mark రీడ్ చేసి పాస్ లేదా ఫెయిల్ అని ప్రింట్ చేయాలి. పాస్ mark 35 గా తీసుకోవాలి.

Solution: ఒక variable లోనికి mark రీడ్ చేసి 35 తో కంపేర్ చేసి ట్రూ అయితే పాస్ అని లేకపోతే ఫెయిల్ అని ప్రింట్ చేద్దాం.

```
#include<stdio.h>
```

```
int main()
    {
    int Marks;
    printf("Enter a student mark");
    scanf("%d", &Marks);
    if(Marks>=35){
    printf("Passed\n");
            }
    else{
    printf("Failed\n");
        }
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రాముని రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



పైన రాసిన ప్రోగ్రాంలో if, else బ్లాక్ లు రెండూ ఒక్క statement మాత్రమే కలిగి వున్నాయి కనుక curly bracets ({, }) తీసేసినా ఫర్వాలేదు. అంతే కాకుండా, if condition ను (Marks<35)గా మార్చి printf statement లను ఎక్స్చేంజ్ చేసినా మనకు రావలసిన రిజల్ట్స్ వస్తాయి.

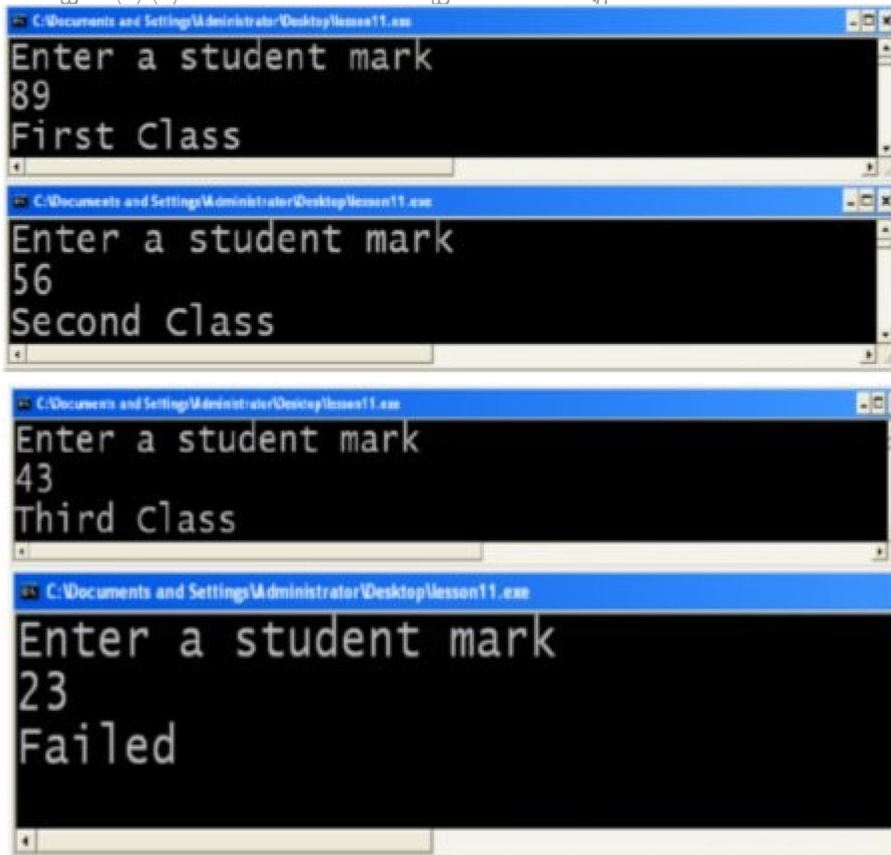
Example 2: ఒక student కు వచ్చిన test mark రీడ్ చేసి ఫస్ట్ క్లాసా, సెకందడ్ క్లాసా, థర్డ్ క్లాసా లేక ఫెయిలా అని ప్రింట్ చేయాలి. ఫస్ట్ క్లాసు, సెకండ్ క్లాసు, థర్డ్ క్లాసు రావడానికి రావలసిన మార్కులు వరుసగా 60, 50, 35.

Solution: ఒక variable లోకి mark రీడ్ చేసి nested if వాడి ముందు 60 తో, తర్వాత 50 తో, ఆ తర్వాత 35 తో కంపేర్ చేస్తే మనకు రావలసిన రిజల్ట్స్ ఈ కింది ప్రోగ్రాం ద్వారా ప్రింట్ చేయవచ్చు.

```
#include<stdio.h>
int main()
    {
    int Marks;
    printf("Enter a student mark\n");
    scanf("%d", &Marks);
```

```
if(Marks>=60)
    printf("First Class\n");
else if(Marks>=50)
    printf("Second Class\n");
else if(Marks>=35)
    printf("Third Class\n");
else
    printf("Failed\n");
return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాముని రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



పైన వాడిన nested if బదులు, ఈ కింద ఇచ్చిన if-else వాడినా సరి పోతుంది.

```
if(Marks>=35)
{
if(Marks>=60)
    printf("First Class\n");
else if(Marks>=50)
    printf("Second Class\n");
else
    printf("Third Class\n");
```

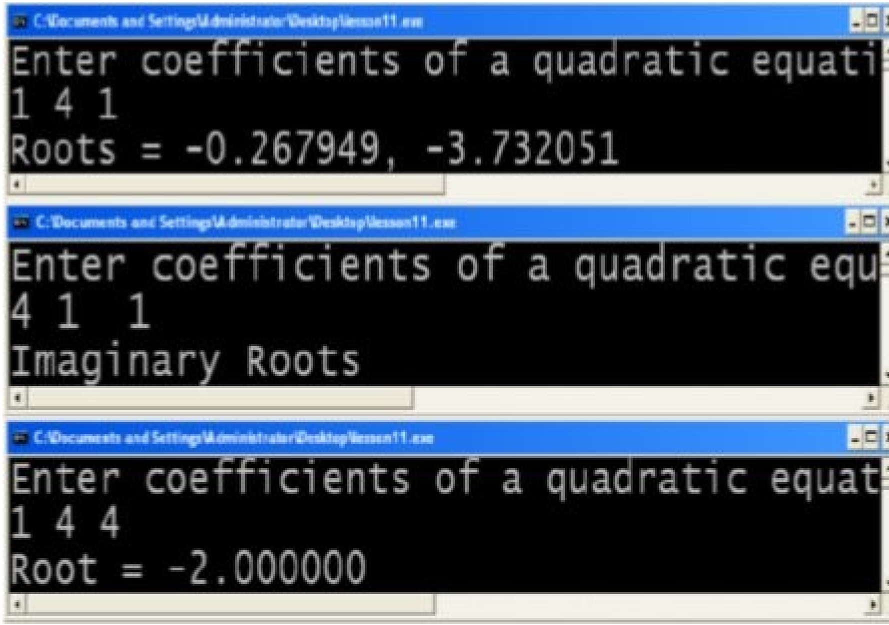
```
    }
    else
    printf("Failed\n");
```

Example 3: ఈ ప్రోగ్రాం quadratic equation యొక్క coefficients రీడ్ చేసి దాని roots ను ప్రింట్ చేస్తుంది.

Solution: Quadratic equation కు మూడు coefficients ఉంటాయి, వాటిని ముందుగా మూడు variables (a,b,c) లోనికి రీడ్ చేస్తాము. ఆ తర్వాత, discriminant function (b2-4ac)ను కాలిక్యులేట్ చేస్తాము. అది కనుక సున్న కన్న తక్కువ అయితే, imaginary roots అని ప్రింట్ చేస్తాము. లేక అదే కనుక సున్నా అయితే ఒక్క root మాత్రమే వుంటుంది, దానిని (-b/2a ను) కాలిక్యులేట్ చేసి ప్రింట్ చేస్తాము. లేకపోతే రెం డు real roots వుంటాయి. వాటిని ((-b+√(b2-4ac))/(2a), (-b-√(b2-4ac))/(2a)) కాలిక్యులేట్ చేసి ప్రింట్ చేస్తాం. మనం square root కాలిక్యులేట్ చెయ్యాలి కనుక math.h ను స్టార్టింగ్ లో include చెయ్యాలి.

```
#include<stdio.h>
#include<math.h>
int main()
    {
    int a,b,c,r1,r2,dis;
    printf("Enter coefficients of a quadratic equation 
ax^2+bx+c=0");
    scanf("%f%f%f", &a, &b, &c);
    dis=b*b-4*a*c;
    if(dis<0) printf("Imaginary Roots\n");
    else if(dis==0)
        {
        r1 = -b/(2*a);
        printf("Root = %f\n", r1);
        }
    else{
        dis=sqrt(dis);
        r1 = (-b+dis)/(2*a);
        r2 = (-b-dis)/(2*a);
        printf("Roots = %f, %f \n", r1, r2);
        }
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



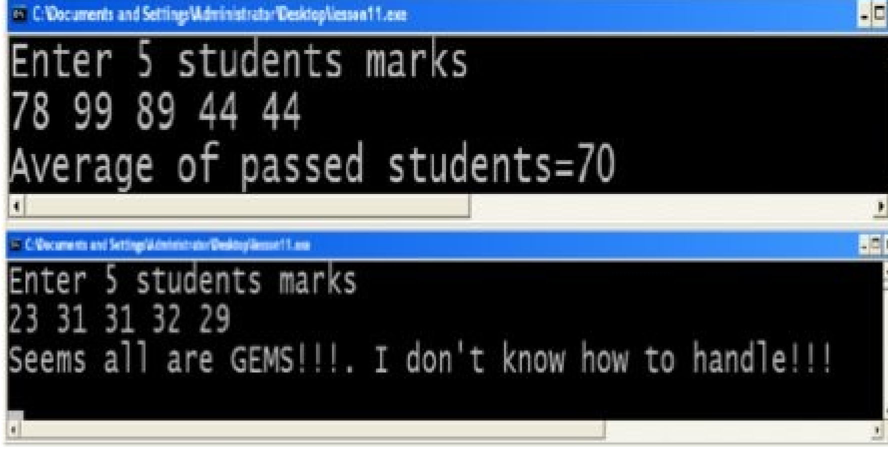
Example 4: ఐదుగురు students యొక్క marks రీడ్ చేసి, పాస్ అయిన వారి యావరేజ్ marks ప్రింట్ చేసే ప్రోగ్రాం ఇది. ఇదే ప్రోగ్రాం ఇంతకు ముందే రాశాం. కానీ ఈ ప్రోగ్రాం రన్ అవుతున్న పుడు run-time error లు ఏవీ రాకుండా ఉండే విధంగా రాసింది.

Solution: ఈ ప్రోగ్రాంను ఇంతకు ముందు రాసినప్పుడు, ఎంతమంది పాస్ అయ్యారు, వారి Total marks కాలిక్యూలేట్ చేసి, ఆ మొత్తాన్ని పాస్ అయిన వారితో divide చేసాం. అప్పుడు “Divided-by-zero” అనే run-time error ను చూశాం. ఈ ప్రోగ్రాము లో అది రాకుండా, ఎంత మంది పాస్ అయ్యారు అనేది సున్నా కాకపోతేనే divide చేస్తున్నాము లేకపోతే ఓ మెసేజి వచ్చేటట్లు if-else ను వాడి రాశాం.

```
#include<stdio.h>
int main()
    {
    int a, b, c, d, e, np, sp, avg;
    printf("Enter 5 students marks");
    scanf("%d%d%d%d%d", &a, &b, &c, &d, &e);
    np=(a>=35)+ (b>=35)+ (c>=35)+ (d>=35)+
(e>=35);
    sp=(a>=35)*a+ (b>=35)*b+ (c>=35)*c+
(d>=35)*d+ (e>=35)*e;
    if(np)
        printf("Average of passed students=%d\n", sp/np);
    else
        printf("Seems all are GEMS!!!. I don't know how to
handle!!!\n");
```

```
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



Example 5: ఒక student 5 test marks ను రీడ్ చేసి ఈ కింద ఇచ్చిన వాటిలో ఎదో ఒక మెసేజి ప్రింట్ చెయ్యాలి.

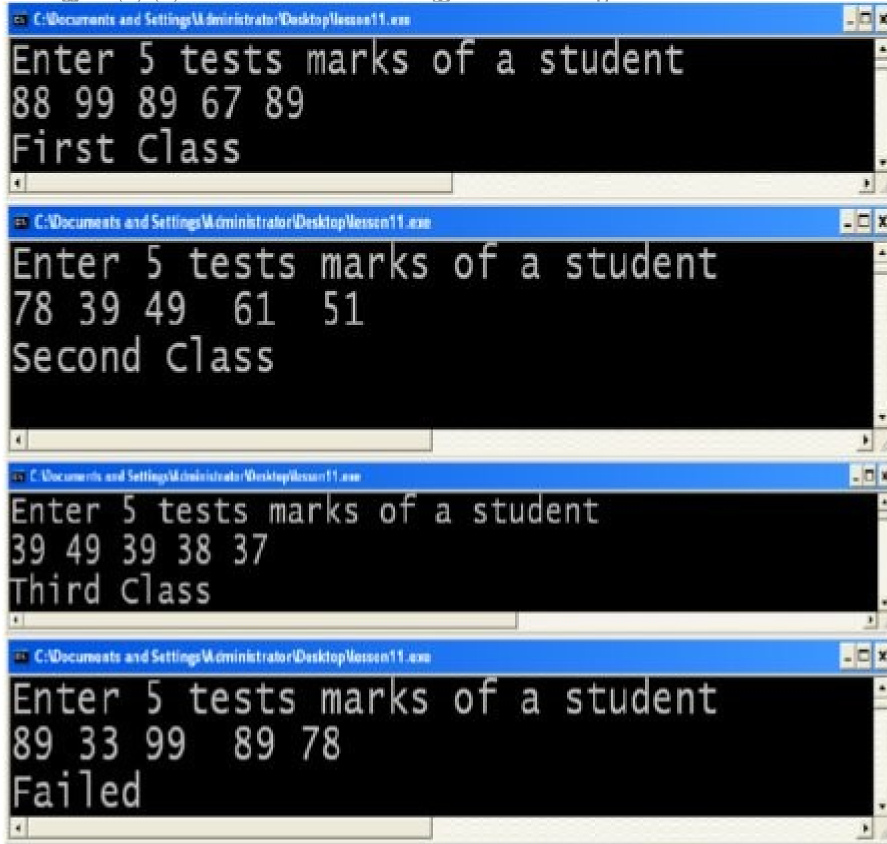
First Class
Second Class
Third Class
Failed

Solution: ముందుగా student కు వచ్చిన 5 test marks ను 5 variables లోకి రీడ్ చేసి, మొత్తం ఎన్నింటిలో పాస్ అయ్యాడో కనుక్కొంటాం. ఇది కనుక 5 అయితే student పాస్ అయినట్లు, లేకపోతే Failed అని ప్రింట్ చేస్తాం. పాస్ అయితే, అప్పుడు మొత్తం మార్కులను కనుక్కొని అది 300 కన్నా ఎక్కువ అయితే First Class అని,250 కన్నా ఎక్కువ అయితే Second Class అని, 175 కన్నా ఎక్కువ అయితే Third Class అని ప్రింట్ చేస్తాం.

```
#include<stdio.h>
int main()
    {
    int a, b, c, d, e, np, s, avg;
    printf(“Enter 5 tests marks of a student\n”);
    scanf(“%d%d%d%d%d”, &a, &b, &c, &d, &e);
    np=(a>=35)+ (b>=35)+ (c>=35)+ (d>=35)+
(e>=35);
    if(np==5)
        {
        s=a+b+c+d+e;
        if(s>=300) printf(“First Class\n”);
        else if(s>=250) printf(“Second Class\n”);
```

```
        else printf("Third Class\n");
        }
    else{
        printf("Failed\n");
        }
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుదు కంప్యూటరు మీద ఇలా వుంటుంది.



పైన ఇచ్చిన ప్రోగ్రాంలో ఈ కింద ఇచ్చిన చిన్నమార్పులను చేయవచ్చు. ఇక్కడ logical AND operator ను వాడి అన్నింటిలో పాస్ అయితే np వాల్యూ 1 లేకపోతే 0 వచ్చేటట్లు రాశాం. దానిని if తో వాడాం.

```
np=(a>=35)&&(b>=35)&&(c>=35)&&(d>=35)&&(e>=35)
;
if(np)
```

2 Labels

లేబిల్ లు ప్రోగ్రాంలో ఎక్కడ కావాలంటే అక్కడ వాడవచ్చు. వీటిని ప్రోగ్రాం యొక్క readability ను లేదా understandability ని పెంచడానికి వాడతాం. ఇవి executable statements కావు. ప్రోగ్రాం రన్ అవుతున్నపుడు లేబిల్ రాగానే program execution next statement కు వెళుతుంది. లేబిల్ పేరులో Upper case, Lower case, 0-9 అక్షరాలు ఉండవచ్చు.బిల్ అనేది, కోలన్, :,

అనే అక్షరం లాస్టుగా వుండే ఓ అక్షరాలు గ్రూప్. ఉదాహరణకు, LOOP: ఒక వాలీడ్ లేబిల్. వీటిని వాడుకొని ఓ ప్రోగ్రామును ఈజీగా అర్ధము అయ్యేటట్లు, ఇలా వ్రాయవచ్చు.

DECLARATIONS:

..................ఇక్కడ variables ను declare చేస్తాం.

....

READING:

.............scanf statements

....

CALCULATIONS:

....

....

PRINTING:

...............printf statements

....

3. The goto statement

ఇది మరొక C control structure, దీనినే unconditional jump అని కుడా అంటారు. (అదే if-else లను అయితే conditional jump లు అని అనవచ్చు. ఎందుకంటే, if expression ట్రూ అయితే ఓ statement గ్రూప్ ను, లేకపోతే వేరేవాటిని రన్ చేస్తాం). దీనిని అంతగా వాడరు. దీని ద్వారా ప్రోగ్రాం execution ను ఒక దగ్గర నుంచి ఇంకో దగ్గరకు మార్చేందుకు వీలు అవుతుంది. దీనికి లేబిల్ ను argument లాగా వాడతారు. ఈ క్రింద ఇచ్చిన ప్రోగ్రాంలో if, goto వాడుకొని ఒక statements గ్రూప్ ను ఎలా రిపీటడ్ గా execute చెయ్యాలో చూపిస్తున్నాం.

Example 6: ఈ ప్రోగ్రాం goto వాడి n students యావరేజ్ ఎలా కనుక్కోవాలో చూపిస్తుంది. ముఖ్యంగా, ఈ క్రింద ఇచ్చిన ప్రోగ్రాంలో if, goto వాడుకొని ఒక statements గ్రూప్ ను ఎలా రిపీటడ్ గా execute చెయ్యాలో చూపిస్తున్నాం. దీనినే if-goto loop అంటారు. ఇక్కడ ముందుగా, ఎంత మంది స్టూడెంట్ లో రీడ్ చేస్తాము (n). తర్వాత, i అనే variable కు సున్నా ఇచ్చి, ఒక student మార్క్స్ రీడ్ చేసినప్పుడల్లా దీనిని ఒకటి పెంచుతాం. దీని వాల్యూ కనుక n కన్నా తక్కువ అయితే మరలా ఇంకో student మార్క్స్ రీడ్ చేస్తాం. ఈ విధంగా అందరి మార్కులను రీడ్ చేసి అప్పుడు యావరేజ్ కనుక్కొంటాము. రీడ్ చేసిన ప్రతీ వాల్యూను s అనే variable కు కలుపుతాం. అలా టోటల్ వస్తుంది. దీనితో యావరేజ్ కనుక్కొంటాము. కింది ప్రోగ్రాం దీనిని వాడుకొని పని చేస్తుంది.

```
#include<stdio.h>
int main()
    {
    int n, i, m, s ;
    printf("Enter number of students");
```

```
scanf("%d", &n);
s=0;
i=0;
LOOPBEGIN: /*(It is a Label)*/
scanf("%d", &m);
s = s + m;
i ++;
if(i<n) goto LOOPBEGIN;
printf("%d, %d\n", s, s/n);
return (0);
}
```

ఈ కింద ఇచ్చిన టేబుల్, పై ప్రోగ్రాం ఎలా పని చేస్తుందో విశదీకరిస్తుంది. ఒక ప్రోగ్రాం ఎలా పని చేస్తుందో తెలుసుకోవచ్చు.ఇలా చేయడాన్ని simulation అంటారు.

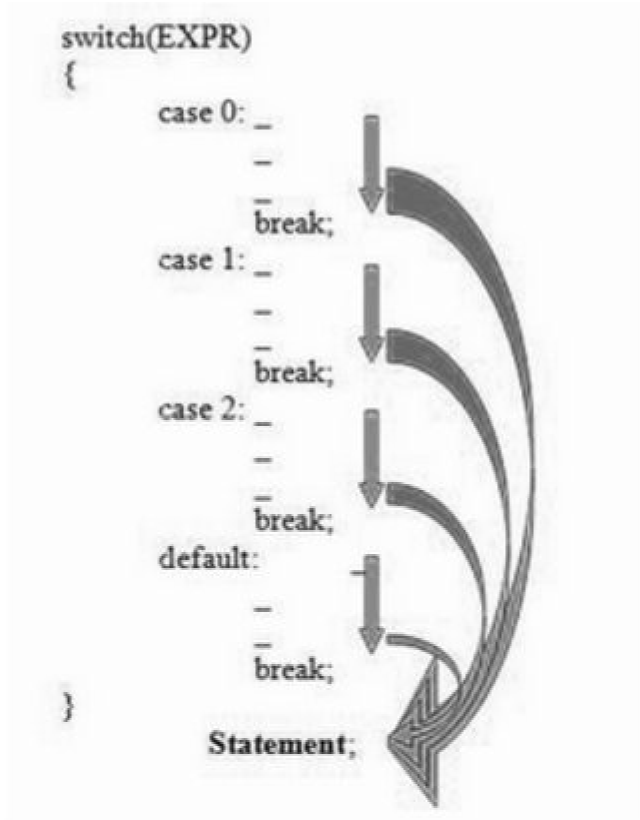
n	i	m	s	Remarks
5				స్టూడెంట్స్ సంఖ్య 5, అని అనుకొందాం.
			0	ముందుగా టోటల్ ను 0 గా తీసికొన్నాం.
	0			i అనే variableకు సున్నా ఇద్చాం.
		70		ఒక స్టూడెంట్ మార్క్స్(70) ఇద్చాం.
			70	మార్క్స్ ను టోటల్ (s)కు కలిపాం.
	1			i variable కు 1 కలిపి n తో పోల్చి చూస్తే, అది n కంటే తక్కువ కనుక program execution LOOPBEGIN అనే లేబిల్ కు వెళ్తుంది.అది లేబిల్ కనుక program execution next statement కు వెళుతుంది.
		90		ఒక స్టూడెంట్ మార్క్స్(90) ఇద్చాం.
			160	మార్క్స్ ను టోటల్ (s) కు కలిపాం.
	2			i variableకు 1 కలిపి n తో కంపేర్ చేస్తే, అది n కన్నా తక్కువ కనుక program execution LOOPBEGIN అనే లేబిల్ కు వెళుతుంది. అది లేబిల్ కనుక program execution next statementకు వెళుతుంది.
		70		ఓ స్టూడెంట్ యొక్క మార్క్స్ (70)ఇద్చాము.
			230	మార్క్స్ ను టోటల్ (s) కు కలిపాము.
	3			i variableకు 1 కలిపి n తో కంపేర్ చేస్తే, అది n కన్నా తక్కువ కనుక program execution LOOPBEGIN అనే లేబిల్ కు వెళుతుంది. అది లేబిల్ కనుక program execution next statementకు వెళుతుంది.
		50		ఓ స్టూడెంట్ మార్క్స్ (50)ఇద్చాం.
			280	మార్క్స్ ను టోటల్ (s) కు కలిపాం.
	4			i variableకు 1 కలిపి n తో కంపేర్ చేస్తే, అది n కన్నా తక్కువ కనుక program execution LOOPBEGIN అనే లేబిల్ కు వెళుతుంది. అది లేబిల్ కనుక program execution next statementకు వెళుతుంది.
		90		ఓ స్టూడెంట్ మార్క్స్ (90) ఇద్చాం.
			370	మార్క్స్ ను టోటల్ (s) కు కలిపాం.
	5			i variableకు 1 కలిపి n తో కంపేర్ చేస్తే, అది n తో సమానం కనుక, యావరేజీ కాలిక్యులేట్ చేసే statementకు program control వెళుతుంది.
				యావరేజ్ ప్రింట్ అవుతుంది.

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుదు కంప్యూటరు మీద ఇలా వుంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of students
5
70
90
70
50
90
Total=370, Average=74
```

4. SWITCH CONSTRUCT

ఇది ఇంకో C లాంగ్వేజ్ control structure. దీనిలో ఒక expression, ఆ expression యొక్క వాల్యూ ఎంత అయినప్పుడు ఏం చెయ్యాలో చెప్పే (వీటినే cases అని అంటారు) statements ఈ కింద విధంగా ఉంటాయి. అంటే ప్రోగ్రాం రన్ అవుతున్నప్పుడు, EXPR వాల్యూ 0 అయితే case 0: తర్వాత వుండే statements (next break వుండేవి) రన్ అవుతాయి. అదే, EXPR వాల్యూ 1 అయితే case 1: తర్వాత వు0 డే statements (next break వు0 డేవి) రన్ అవుతాయి. అదే, EXPR వాల్యూ 2 అయితే case 2: తర్వాత ఉండే statements (next break ఉండేవి) రన్ అవుతాయి. అలా కాకుండా, EXPR వాల్యూఇంకేదయినా అయితే, default: తర్వాత ఉండే statements (next break ఉండేవి) రన్ అవుతాయి. దీనినే default clause అని అంటారు. ఈ default clause లేకుండా కూడా switch construct ను రాయవచ్చు. ఈ switch construct రాసేటప్పుడు, case అనే word తర్వాత integer constants, character constants ను మాత్రమే వాడాలి.



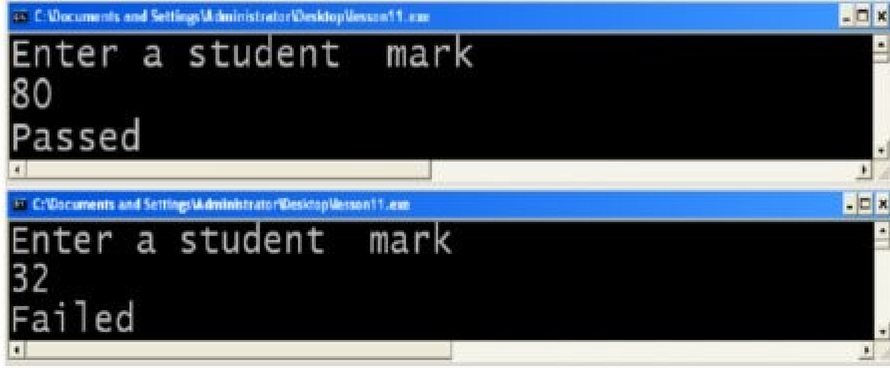
Example 7: ఈ ప్రోగ్రాం ఒక student marks రీడ్ చేసి పాసో, ఫెయిలో ప్రింట్ చేస్తుంది. ఇది ఇంతకు ముందు if-else లు వాడి రాసిందే. కానీ ఇక్కడ, switch వాడి చేస్తున్నాము.

Solution: రీడ్ చేసిన student mark ను 35 తో కంపేర్ చేసేది switch లో వాడుతున్న expression. ఇది 1 అయితే (అంటే పాస్ అయితే), case 1 తో మాచ్ అవుతుంది. కనుక తర్వాత ఉండే statements రన్ అవతాయి. లేకుంటే, case 0 తర్వాత వుండే statements రన్ అవతాయి. ఉదాహరణకు, 75 ఇస్తే case 1 తో మాచ్ అవుతుంది. కనుక case 1 తర్వాత ఉండే statements రన్ అవతాయి, 25 ఇస్తే case 0 తర్వాత ఉండే statements రన్ అవతాయి.

```
#include<stdio.h>
int main()
    {
    int n;
    printf("Enter a student mark");
    scanf("%d", &n);
    switch (n>=35){
        case 0:
            printf("Failed\n");
            break;
        case 1:
```

```
            printf("Passed\n");
            break;
        }
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రాముకు రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



Example 8: ఒక student కు వచ్చిన test marks రీడ్ చేసి ఈ కింద ఇచ్చిన వాటిలో ఎదో ఒక మెసేజి ప్రింట్ చెయ్యాలి.

First Class
Second Class
Third Class
Failed

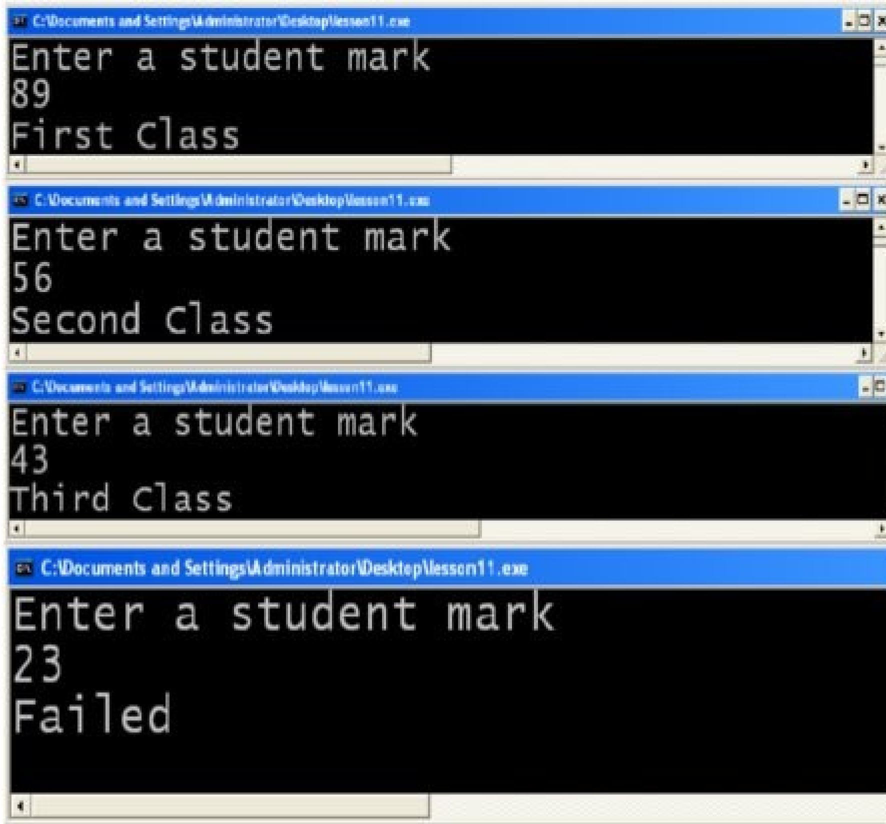
Solution: ముందు ఒక switch ని వాడుతూ marks ను 35 తో కంపేర్ చేస్తాం. పాస్ అయితే (case 1), ఇంకో switch వాడి Class డిసైడ్ చేస్తాం. రెండో switch లో expression ను n/10 గా వాడాము. ఈ కింది టేబిల్, marks ఎంత వుంటే n/10 ఎంత చూపిస్తుంది అనే దానిని చూపిస్తుంది. దీనిపై ఆధారపడి, switch case లను design చేశాము.

marks	n/10
35-39	3
40-49	4
50-59	5
>=60	6,7,8,9,10

```
#include<stdio.h>
int main()
    {
    int n;
    printf("Enter a student mark\n");
    scanf("%d", &n);
    switch (n>=35){
```

```
        case 0:
            printf("Failed\n");
            break;
        case 1:
            switch(n/10){
            case 3:
            case 4: printf("Third Class\n");
                    break;
            case 5: printf("Second Class\n");
                    break;
            default: printf("First Class\n");
                    break;
            }
            break;
        }
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



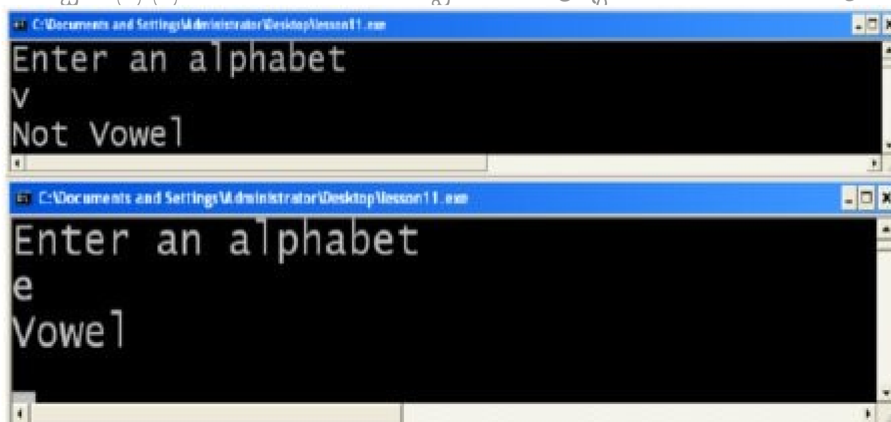
ఇక్కడ case 3, case 4 లలో ఒకే statements అనగా printf("Third Class\n");, రన్ అవుతాయి. ఇలాగా case లను గ్రూప్ చెయ్యవచ్చు. అనగా n/10 వాల్యూ 3 అయినా 4 అయినా సేమ్ పని చేస్తుంది.

Example 9: ఈ ప్రోగ్రాం ఒక character ను రీడ్ చేసి అది Vowel అవునా కాదా అని ప్రింట్ చేస్తుంది.

Solution: ఒక character ను v అనే variable లోకి రీడ్ చేసి దానిని switch expression లా వాడుతున్నాం. అది కనుక 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u', లలో ఏదో ఒకటి అయితే vowel అని లేకుంటే కాదని ప్రింట్ చేస్తున్నాం.

```
#include<stdio.h>
int main()
    {
    char v;
    printf("Enter an alphabet\n");
    scanf("%c", &v);
    switch (v){
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U':
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u': printf("Vowel\n");
                   break;
        default: printf("Not Vowel\n"); break;
      }
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రాముని రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



ఈ కింద కొద్ది మార్పుతో మరో ప్రోగ్రాంను చేసాం. ఇక్కడ ముందుగా రీడ్ చేసిన దానిని అప్పర్ లోకి మార్చి, ‘A’, ‘E’, ‘I’, ‘O’, ‘U’, మాత్రమే మ్యాచ్ చేస్తున్నాం. కావున ఐదు కేస్ లు మాత్రమే ఇక్కడ వాడాం .

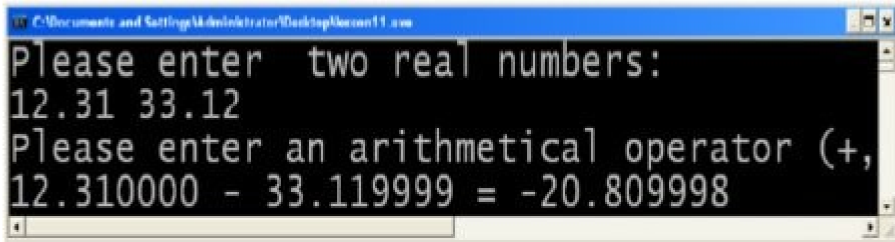
```
#include<stdio.h>
#include<ctype.h>
int main()
     {
    char v;
    printf("Enter an alphabet");
    scanf("%c", &v);
    switch (toupper(v)){
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U': printf("Vowel\n");
                    break;
        default: printf("Not Vowel\n");
                  break;
        }
    return (0);
    }
```

Example 10: ఈ కింది ప్రోగ్రాం రెండు వాల్యూస్ ను, మరియు operator ను రీడ్ చేసి, ఆ operator ను operands మధ్యలో అప్లై చేసి రిజల్ట్ ప్రింట్ చేస్తుంది.

```
#include <stdio.h>
int main( )
    {
    double n1, n2, res;
    char op;
    printf("Please enter real numbers: ");
    scanf("%lf%lf", &n1, &n2);
    printf("Please enter an arithmetical operator (+, -, *
or /): ");
    scanf(" %c", &op);
    switch(op) {
    case '+':
        res = n1+n2;
        break;
```

```
    case '-':
        res = n1-n2;
        break;
    case '*':
        res = n1*n2;
        break;
    case '/': /* We're not checking for division by zero for clarity... */
        res = n1/n2;
        break;
    default:
        printf("%c is an invalid arithmetical operator!\n", op);
        return 1;
    }
printf("%g %c %g = %g\n", n1, op, n2, res);
return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాముకు రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.



5. Conclusions

ఈ చాప్టరులో if-else, goto, switch అనే control structures గురించి నేర్చుకొన్నాం. ప్రతీ దానితో రేడీ టు రన్ ప్రోగ్రాంలను కూడా ఇచ్చాం.

'సీ' లో లూప్ లు ఏం చేస్తాయి?

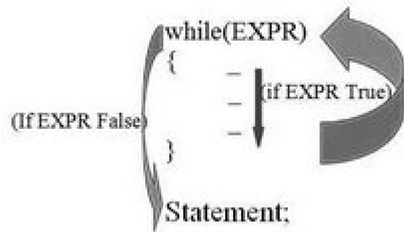
ఒక statements గ్రూప్ ను రిపీటెడ్ గా రన్ చేసేందుకు loop లను వాడతారు. loop లు అనేవి high level programming లాంగ్వేజీలో ఉండే ముఖ్యమైన ఫీచర్. C లాంగ్వేజి లో మూడు రకాల loop లు ఉన్నాయి. వాటి గురించి ఈ పాఠంలో నేర్చుకుందాం.

1. while loop
2. do-while loop
3. for loop

1 The while loop

ఈ కింద ఇచ్చిన బొమ్మ while loop ఎలా పని చేస్తుందో చూపిస్తుంది. EXPR విలువ

true గా ఉన్నంతవరకు while బ్లాక్ ({ మరియు } మధ్యలో ఉన్న statements) లో ఉన్న statements రిపీటెడ్ గా execute చేస్తుంది. EXPR విలువ false అయినప్పుడు లేదా while బ్లాక్ లోపల break statement ను execute చేసినప్పుడు program control loop లో నుంచి బయటకు, అంటే Statement కు వస్తుంది.



ముఖ్యంగా while loop బిహేవియర్ ను, EXPR లో వాడే variables (వీటిని control variables అని అంటారు), వాటి మొదటి (initial) విలువస్, లూప్ లోపల వాటి విలువ మార్చే statements (వాటినే Modifier లు అంటారు), కలసి control చేస్తాయి. ఇంకా, while loop బ్లాక్ లో ఒక్క statement మాత్రమే వుంటే, {, మరియు }లను తీసివేయవచ్చు.

Example 1: ఈ కింది ప్రోగ్రాం while loop ఎలా పని చేస్తుందో విశదీకరిస్తుంది.

```
#include<stdio.h>
int main()
{
    int i, n; (here, "i, n" are loop control variables)
    printf("Enter an integer\n");
    scanf("%d", &n);
    i=0; (Initialization Statement)
    while(i<n) (Condition)
        { _
        _
        i++; (Modifier)
        }
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంలో, లూప్ ఎలా పని చేస్తుందో ఈ కింది పట్టిక ద్వారా step-by-step పరిశీలించి తెలుసుకుందాం.

n	i	రిమార్క్స్
5		n కు 5 విలువ scanf ద్వారా ఇద్దామనుకుందాం.
	0	i అనే variableకు సున్నా ఇద్దాం (Initialization Statement).
		ప్రస్తుతం, i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
	1	i variableకు 1 కలుపుతాం. తర్వాత, Program execution, condition చెకింగ్ కు వెళుతుంది.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
	2	i variableకు 1 కలుపుతాం. తర్వాత, Program execution, condition చెకింగ్ కు వెళుతుంది.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
	3	i variableకు 1 కలుపుతాం. తర్వాత, Program execution, condition చెకింగ్ కు వెళుతుంది.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
	4	i variableకు 1 కలుపుతాం. తర్వాత, Program execution, condition చెకింగ్ కు వెళుతుంది.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
	5	i variableకు 1 కలుపుతాం. తర్వాత, Program execution, condition చెకింగ్ కు వెళుతుంది.
		i విలువ nకు సమానం కాబట్టి program execution లూప్ లోనుంచి బయటకు వెళుతుంది.

ప్రోగ్రాంలో ఉన్న లూప్ ను పరిశీలించడం ద్వారా , మనకు కొన్ని విషయాలు తెలిసాయి. అవి , ఆ లూప్ లో ఉన్న statements అయిదు సార్లు రన్ అయ్యాయి; అది కూడా n విలువ 5 కాబట్టి మొత్తంగా ఈ లూప్ n సార్లు రన్ అవుతుందని చెప్పవచ్చు. అంతేకాక, i++ లేకపోయినా, లేదా i++ స్థానంలో i--వాడినా, లూప్ నిరంతరం (infinite) రన్ అయ్యేది . అలాగే, ఈ కింద చూపిన విధంగా, బ్రాకెట్ తర్వాత ; వాడినా లూప్ నిరంతరం (infinite) గా రన్ అవుతుంది. ఎందుకంటే ; ను while loop బ్లాక్ లాగా తీసుకుంటుంది. దానినే మనం do nothing బ్లాక్ అని కూడా అనవచ్చు. అంటే, i<n అయ్యేంతవరకూ ఏమీ చేయవద్దు (do nothing) అని. దీనిని operating systems సబ్జెక్టులో busy wait అని అంటారు.

```
while (i<n);
```

మరియూ లూప్ control variables ను initialize చెయ్యకపోతే, లూప్ ఒక్కోసారి రన్ అవుతుంది, కొన్ని సార్లు మనం అనుకొన్న దానికంటే ఎక్కువ సార్లు రన్ అవుతుంది, కొన్ని సార్లు అసలు రన్ కూడా అవ్వదు.

Example 2: ఈ ప్రోగ్రాం తరగతిలో ఎంత మంది విద్యార్థులు ఉన్నారో రీడ్ చేసి, వారి మార్కులను రీడ్ చేసి సరాసరి ప్రింట్ చేస్తుంది.

Solution: పైన వివరించిన while loop నే ఈ example లో కూడా వాడుతున్నాం. అంటే n సార్లు, ఎంతమంది విద్యార్థులు ఉంటే అన్ని సార్లు రన్ అయ్యే లూప్ వాడి, ప్రతీ సారీ ఒక విద్యార్థి మార్కులను రీడ్ చేసి మొత్తాన్ని కనుక్కొని ఆ తర్వాత సరాసరి కనుక్కొంటుంది.

```
#include<stdio.h>
int main()
{
    int n, i, m, s;
    printf("Enter how students average you want\n");
    scanf("%d", &n);
    i=0;
    s=0;
    while(i<n){
    printf("Enter a student mark\n");
    scanf("%d", &m);
    s=s+m;
    i++;
    }
   printf("Average= %d\n", s/n);
   return 0;
}
```

ఈ కింది పట్టిక ద్వారా ఈ ప్రోగ్రాంను step-by-step పరిశీలించి చేస్తే మనకు ఇది ఎలా పనిచేస్తుందో అర్థమవుతుంది.

n	i	m	s	Remarks
3				n కు 3 విలువ scanf ద్వారా ఇచ్చామనుకుందాం.
	0		0	i, s లకు సున్నాలు ముందుగా ఇచ్చాం. ఇప్పుడు, i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
		90		ఒక విద్యార్థి మార్కులు 90 గా ఇచ్చామనుకుందాం.
			90	90 ను మొత్తానికి అంటే s variableకు add చేస్తున్నాం.
	1			i విలువ ను ఒకటి పెంచుతున్నాం. ఇప్పుడు, i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
		80		ఇంకో విద్యార్థి మార్కులను 80 గా ఇచ్చామనుకుందాం.
			170	80 ను మొత్తానికి అంటే s variableకు add చేస్తున్నాము.
	2			i విలువ ను ఒకటి పెంచుతున్నాం. ఇప్పుడు, iవిలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది.
		90		మరొక విద్యార్థి మార్కులను 90 గా ఇచ్చామనుకుందాం.
			260	90 ను మొత్తానికి అనగా s variableకు add చేస్తున్నాము.
	3			i విలువను ఒకటి పెంచుతున్నాం. ఇప్పుడు, i విలువ n తో సమానం కాబట్టి program execution లూప్ బయటకు వెళుతుంది.
				సరాసరిని ప్రింట్ చేస్తుంది.

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter how students average you want
3
Enter a student mark
90
Enter a student mark
80
Enter a student mark
90
Average= 86
```

Example 3: ఈ ప్రోగ్రాం తరగతిలో ఎంత మంది విద్యార్థులు ఉన్నారో రీడ్ చేసి, వారి మార్కులను రీడ్ చేసి ఉత్తీర్ణులైన విద్యార్థి ల మార్కుల సరాసరిను ప్రింట్ చేస్తుంది. పాస్ కావాలంటే 35 మార్కులు రావాలి.

Solution: పైన డిస్కస్ చేసిన while loop నే ఈ example లో కూడా వాడుతున్నాం. అంటే ఎంతమంది విద్యార్థులు ఉంటే అన్ని సార్లు రన్ అయ్యే లూప్ వాడి, ప్రతీ సారీ ఒక విద్యార్థి మార్కులను రీడ్ చేసి, దానిని 35 తో పోల్చి ఎక్కువ అయితే మొత్తం(s)కు add చేసి np ను 1 పెంచుతాం. అంటే ఉత్తీర్ణులైన వారి మొత్తంమార్క్స్, ఎంత మంది పాస్ అయ్యారో కనుక్కొంటాం. Program execution లూప్ బయటకు వచ్చిన తర్వాత, సరాసరి కనుక్కొని ప్రింట్ చేస్తాం. సరాసరి

కనుక్కొనేటప్పుడు, Divided by Zero అనే error రాకుండా, np విలువ సున్నా కాకపోతేనే సరాసరి ను ప్రింట్ చేస్తున్నాం, లేకపోతే ఒక మెసేజీ వచ్చేవిధంగా చేస్తున్నాం.

```
#include<stdio.h>
int main()
{
    int n, i, m, sp = 0, np = 0;
    printf("Enter Number of Students\n");
    scanf("%d", &n);
    i=0;
    while(i<n)
        {
        printf("Enter next student mark\n");
        scanf("%d", &m);
        if(m>=35){
        sp += m;
        np++;
        }
    i++;
    }
    if(np)
    printf("Average of Passed Students=%d\n", sp/np);
    else
    printf("Seems All GEMS. Don't Know How to
Handle!!!!\n");
    return (0);
}
```

ఈ ప్రోగ్రాంకు మనం పైన చేసిన విధంగా పరిశీలించి పట్టికను తయారు చెయ్యండి.
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter Number of Students
3
Enter next student mark
90
Enter next student mark
8
Enter next student mark
90
Average of Passed Students=90
```

```
Enter Number of Students
5
Enter next student mark
31
Enter next student mark
23
Enter next student mark
32
Enter next student mark
34
Enter next student mark
19
Seems All GEMS. Don't Know How to Handle!!!!
```

Example 4: ఈ ప్రోగ్రాం ఒక విద్యార్థి మార్కులను n టెస్ట్ లలో రీడ్ చేసి అతని క్లాస్ ను ఈ కింద ఇచ్చిన నియమాల ప్రకారం ప్రింట్ చేస్తుంది.

Average Marks	Class
>=60	First Class
50-50	Second Class
35-49	Third Class
<35	Failed

Solution: పైన వివరించిన while loop నే ఈ example లో కూడా వాడుతున్నాం. అంటే ఎన్ని టెస్టు లు ఉంటే అన్ని సార్లు రన్ అయ్యే లూప్ వాడి, ప్రతీ సారీ ఒక టెస్టు మార్క్ రీడ్ చేసి టోటల్ కు add చేస్తున్నాం. అంతే కాకుండా, ఆ టెస్ట్ లో ఉత్తీర్ణులైతే, అంటే 35 కంటే ఎక్కువ వస్తే, np విలువను ఒకటి పెంచుతున్నాం. ప్రతీ టెస్ట్ లో ఉత్తీర్ణులైతేనే ఏదయినా క్లాస్ ఇస్తాం. ఈ np value n తో సమానం అయితే, అన్ని టెస్ట్ లలో ఉత్తీర్ణులైనట్లు. ఆ తర్వాత, యావరేజ్ ను తీసుకొని క్లాస్ ప్రింట్ చేస్తాం.

```
#include<stdio.h>
int main()
{
    int n, i, m, s=0,np=0;
    printf("Enter number of tests\n");
    scanf("%d", &n);
```

```
    i=0;
    while(i<n){
    printf("Enter next test mark\n");
    scanf("%d", &m);
    s+=m;
    if(m>=35)np++;
    i++;
    }
    if(np==5){
    s=s/n;
    if(s>=60)
    printf("First class\n");
    else if(s>=50)
    printf("Second class\n");
    else
    printf("Third class\n");
    }
    else
    printf("Failed\n");
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of tests
5
Enter next test mark
90
Enter next test mark
89
Enter next test mark
90
Enter next test mark
78
Enter next test mark
98
First class
```

```
Enter number of tests
5
Enter next test mark
89
Enter next test mark
90
Enter next test mark
30
Enter next test mark
90
Enter next test mark
89
Failed
```

Example 5: ఈ ప్రోగ్రాం తరగతిలో ఎంత మంది విద్యార్థులు ఉన్నారో రీడ్ చేసి, వారి మార్కులను రీడ్ చేసి సరాసరి, standard Deviation ప్రింట్ చేస్తుంది.

Solution: విద్యార్థి మార్కులను x1, x2, x3,….,xn అనుకుంటే, mean, standard deviation లను కనుకోవడానికి ఈ కింది సూత్రాలను వాడతాం. పైన వివరించిన while loop నే ఈ example లో కూడా వాడుతున్నాం. అంటే ఎంతమంది విద్యార్థులు ఉంటే అన్ని సార్లు రన్ అయ్యే లూప్ వాడి, ప్రతీ సారీ ఒక విద్యార్థి మార్క్ రీడ్ చేసి దానిని మొత్తం(s) కు, దాని స్క్వేర్ ను s కు add చేస్తున్నాము. లూప్ లోనుంచి బయటకు వచ్చిన తర్వాత, సరాసరి, standard Deviation కాలిక్యులేట్ చేసి ప్రింట్ చేస్తున్నాం.

```
#include<stdio.h>
#include<math.h>
int main()
{
    float s=0, ss = 0;
    int n, i, m;
    printf("Enter number of Students\n");
    scanf("%d", &n);
```

```
    i = 0;
    while(i<n){
    printf("Enter a student mark\n");
    scanf("%d", &m);
    s = s + m;
    ss = ss + m*m;
    i++;
    }
    s=s/n;
    ss=sqrt((ss/n)-(s*s));
    printf("Mean=%f Standard Deviation=%f\n", s, ss);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of Students
5
Enter a student mark
90
Enter a student mark
90
Enter a student mark
90
Enter a student mark
90
Enter a student mark
90
Mean=90.000000 Standard Deviation=0.000000
```

```
Enter number of Students
5
Enter a student mark
98
Enter a student mark
34
Enter a student mark
89
Enter a student mark
36
Enter a student mark
87
Mean=68.800003 Standard Deviation=27.852461
```

Example 6: ఈ ప్రోగ్రాం తరగతిలో ఎంత మంది విద్యార్థులు ఉన్నారో రీడ్ చేసి, వారి మార్కులను రీడ్ చేసి, largest and the second largest మార్కులను ప్రింట్ చేయడానికి ఉద్దేశించినది.

Solution: ఇందులో కూడా విద్యార్థులు ఎంత మంది ఉంటే అన్ని సార్లు రన్ అయ్యే లూప్ వాడి, ప్రతీ సారి ఒక విద్యార్థి మార్క్(m) రీడ్ చేస్తాం. ముందుగా m max1,

max2 అనే రెండింటిని తీసుకొని max1 కు ముందుగా సున్నా ఇస్తాం. మనం తీసుకొన్న విద్యార్థి మార్క్(m) max1 కంటే ఎక్కువ అయితే max1 విలువను max2 కు ఇచ్చి, ప్రస్తుతం తీసుకొన్న విద్యార్థి మార్క్(m)ను max1 ఇస్తాం. లేకపోతే ప్రస్తుతం తీసుకొన్న విద్యార్థి మార్క్(m) max2 కంటే ఎక్కువ అయితే దానిని max2 కు ఇస్తాం. తీసుకొన్న ప్రతీ విద్యార్థి మార్కుల తో ఈ విధంగా చేస్తాం. లూప్ లోనుంచి బయటకి వచ్చిన తర్వాత, max1, max2 లను ప్రింట్ చేస్తాం.

```
#include<stdio.h>
int main()
{
    int n,i,m,max1=0,max2;
    printf(“Enter number of students\n”);
    scanf(“%d”, &n);
    i = 0;
    while(i<n){
    printf(“Enter next student mark\n”);
    scanf(“%d”, &m);
    if(m>max1){
    max2 = max1;
    max1=m;
    }
    else if(m>max2&&m<max1) max2=m;
    i++;
    }
    printf(“Highest=%d Second Highest=%d\n”, max1,
max2);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.

```
Enter number of students
5
Enter next student mark
98
Enter next student mark
31
Enter next student mark
78
Enter next student mark
99
Enter next student mark
15
Highest=99 Second Highest=98
```

Example 7: ఈ ప్రోగ్రాం ఒక nteger ‘n’ ను రీడ్ చేసి దానికి సంబంధించిన ఎక్కము ప్రింట్ చేస్తుంది. ఉదాహరణకు, 7 ఇస్తే, ఏడో ఎక్కము కింద ఇచ్చిన విధముగా ప్రింట్ చెయ్యాలి.

```
7 X 1=7
7 X 2=14
7 X 3=21
--------
--------
7 X 20=140
```

Solution: పైన రావల్సిన ఫలితాన్ని చూస్తే మనకు ఒక విషయం అర్థమవుతుంది. అదేమిటంటే, 1 నుంచి 20 దాకా మారే ఒక లూప్ కావాలని. దానిని వాడుకొని కింద ఇచ్చిన ప్రోగ్రాం రాద్దాం.

```
#include<stdio.h>
int main()
{
    int n, i;
    printf("Enter an integer whose multiplication table is
needed\n");
    scanf("%d", &n);
    i=1;
    while(i<=20){
    printf("%d X %d = %d\n", n, i, n*i);
    i++;
    }
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా వుంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter an integer whose multiplication table is needed
7
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
7 X 10 = 70
7 X 11 = 77
7 X 12 = 84
7 X 13 = 91
7 X 14 = 98
7 X 15 = 105
7 X 16 = 112
7 X 17 = 119
7 X 18 = 126
7 X 19 = 133
7 X 20 = 140
```

Example 8: ఈ ప్రోగ్రాం ఒక integer (n)ను రీడ్ చేసి దాని factorial విలువను లెక్కిస్తుంది.

Solution: ఒక integer (n) factorial value అనగా: 1x2x3x…xn. అంటే, ఇక్కడ కూడా 1 నుంచి n దాకా మారే ఒక లూప్ కావాలి. దానిని వాడుకొని కింద ఇచ్చిన ప్రోగ్రాం రాద్దాం.

```
#include<stdio.h>
int main()
{
    int n, i, f ;
    printf("Enter an integer\n");
    scanf("%d", &n);
    f=1;
    i=1;
    while(i<=n){
    f = f*i;
    i++;
    }
    printf("Factorial Value of %d=%d\n",n, f);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



Example 9: ఈ ప్రోగ్రాం ఒక integer (n)ను రీడ్ చేసి దానిలో ఉండే సంఖ్యల మొత్తంను లెక్కిస్తుంది.

Solution: ఒర పూర్ణ సంఖ్యకు 10 తో modulus operator అప్లై చేస్తే ఆఖరి సంఖ్య వస్తుంది, అదే 10 తో divide చేస్తే ఆఖరి సంఖ్య తీసేయగా మిగిలిన పూర్ణసంఖ్య వస్తుంది. ఉదాహరణకు, 1889 కు 10 తో modulus operator అప్లై చేస్తే 9 వస్తుంది, అదే 10 తో divide చేస్తే 188 వస్తుంది. ఇదే లాజిక్ ను 188 మీద అప్లై చేస్తే 8 మరియు 18 వస్తాయి. ఇలా, ప్రతీసారి ఓ డిజిట్ ను లాగి మొత్తానికి కలుపుతాం. ఇక్కడ, while(n) అనే లూప్ యొక్క కండిషను, విలువ సున్నా అయ్యేంత వరకు ట్రూ అవుతుంది.

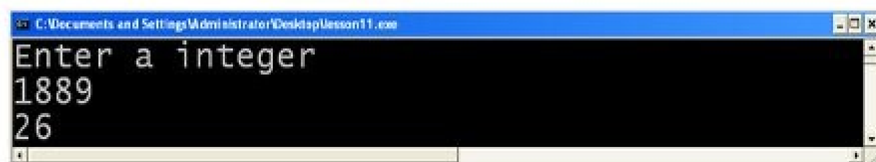
```
#include<stdio.h>
int main()
{
```

```
    int n, s=0, dig;
    printf("Enter a integer\n");
    scanf("%d", &n);
    while(n){
    dig = n%10;
    s = s + dig;
    n = n/10;
    }
    printf("%d\n", s);
    return (0);
}
```

ఈ కింది పట్టిక ద్వారా ఈ ప్రోగ్రాంను step-by-step పరిశీలించి చేస్తే మనకు ఇది ఎలా పనిచేస్తుందో అర్థమవుతుంది.

n	dig	s	Remarks
1889		0	లూప్ కండిషన్ ట్రూ, కాబట్టి కంట్రోలు లూప్ లోపలికి వెళుతుంది.
	9		లాస్ట్ డిజిట్ 9 వస్తుంది
		9	9 s కు add అవుతుంది.
188			**10 చేత divide అయిన తర్వాత, n 188 అవుతుంది.** **ఇప్పుడు,** లూప్ కండిషన్ ట్రూ, కాబట్టి కంట్రోలు లూప్ లోపలికి వెళుతుంది.
	8		లాస్ట్ డిజిట్ 8 వస్తుంది.
		17	8 s కు add అవుతుంది.
18			**10 చేత divide అయిన తర్వాత, n 18 అవుతుంది.** **ఇప్పుడు,** లూప్ కండిషన్ ట్రూ, కాబట్టి కంట్రోల్ లూప్ లోపలికి వెళుతుంది.
	8		లాస్ట్ డిజిట్ 8 వస్తుంది.
		25	8 s కు add అవుతుంది.
1			**10 చేత divide అయిన తర్వాత, n 1 అవుతుంది.** **ఇప్పుడు,** లూప్ కండిషన్ ట్రూ, కాబట్టి కంట్రోల్ లూప్ లోపలికి వెళుతుంది.
	1		లాస్ట్ డిజిట్ 1 వస్తుంది.
		26	1 s కు add అవుతుంది.
0			**10 చేత divide అయిన తర్వాత, n 0 అవుతుంది.** **ఇప్పుడు,** లూప్ కండిషన్ ఫాల్స్ కాబట్టి లూప్ లో నుంచి బయటకు వచ్చి s విలువ ప్రింట్ చేస్తాం.

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



1.1 The break statement

ఈ break statement ను వాడి మనం ఎప్పుడైనా లూప్ లోనుంచి బయటకు రావచ్చు. దీనిని వేరే లూప్ లలో కూడా వాడవచ్చు.

Example 10: ఈ ప్రోగ్రాం ఒక integer (n)ను రీడ్ చేసి అది ప్రధాన సంఖ్యా (Prime number) కాదా ప్రింట్ చేస్తుంది.

Solution: ప్రధాన సంఖ్య అనాలి అంటే అది 1 మరియూ దాని చేత మాత్రమే divide అవ్వాలి. వేరే ఏ నంబరూ దానిని divide చెయ్యకూడదు. అందుకని 2 నుంచి మనం ఇచ్చిన సంఖ్య లోపల ఉండే ప్రతీ సహజ సంఖ్యను తీసుకొని అది ఇచ్చిన సంఖ్యను divide చేస్తుందా అని ఓ లూప్ ద్వారా టెస్ట్ చేస్తాం. ఏదయినా సహజసంఖ్య మనం ఇచ్చిన సంఖ్య divide చేస్తే break statement ను వాడి లూప్ లోనుంచి బయటకు వచ్చి, అది ప్రధాన సంఖ్య కాదు అని ప్రింట్ చేస్తాం. ఆదే మామూలుగా లూప్ లోనుంచి బయటకు వస్తే అది ప్రధానసంఖ్య (Prime number).

```
#include<stdio.h>
int main()
{
    int i, n;
    printf("Enter an integer\n");
    scanf("%d", &n);
    i=2;
    while(i<n){
    if(n%i==0) break;
    i++;
    }
    (i==n)?printf("ప్రధాన Number\n"): printf("Not a ప్రధాన
number\n");
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



1.2. The exit function

ఇది standard library లోని ఒక function. దీనిని call చేస్తే program

లోనుంచి బయటకు వస్తాం. దానినే ప్రోగ్రాం టెర్మినేట్ చేయడం అని అనవచ్చు. దీనికి మనం ఒక integer ను argument లాగా ఇవ్వాలి.
పై ప్రోగ్రాంను ఈ exit function వాడి కింది విధముగా రాయవచ్చు.

```
#include<stdio.h>
int main()
{
    int i, n;
    printf("Enter an integer\n");
    scanf("%d", &n);
    i=2;
    while(i<n)
    {
    if(n%i==0)
    {
    printf("Not a ప్రధాన number\n");
    exit(-1);
    }
    i++;
    }
    printf("ప్రధాన number\n");
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



1.3 Continue Statement

ఈ "continue" statement ను while loop లో వాడితే దీని తర్వాత ఉండే statements ను execute చేయకుండా లూప్ కంట్రోలు కండిషన్ చెకింగుకు వెళుతుంది. దీనిని వేరే లూప్ లలో కూడా వాడవచ్చు.

Example 11: ఈ ప్రోగ్రాంలో "continue" statement ను while loop లో వాడితే ఏమవుతుందో చూద్దాం. ఇక్కడ ఒక infinite loop(while(1)), అనే దానిని వాడాం. దీనిలో ప్రతీ సారి ఓ సంఖ్య రీడ్ అవుతుంది. కాబట్టి 0 అయితే లూప్ లోనుంచి break statement ను వాడి బయటకు వచ్చేస్తాం. ఇది కాబట్టి బేసి సంఖ్య

అయితే "continue" statement ను వాడి next iteration కు వెళుతుంది. లేకపోతే ఇచ్చిన సంఖ్యను s కు add అవుతుంది. అలా ఈ ప్రోగ్రాం 0 ఇచ్చే లోపల మనం ఇచ్చిన సరి సంఖ్యల మొత్తాన్ని కనుక్కొని, వాటి సరాసరిని ప్రింట్ చేస్తుంది.

```
#include<stdio.h>
int main()
{
    int n=0, s=0, m;
    while(1)
    {
    printf("Enter a student mark\n");
    scanf("%d", &m);
    if(m==0) break;
    else if(m%2) continue;
    s=s+m;
    n++;
    }
    printf("%d\n", s/n);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter a student mark
89
Enter a student mark
80
Enter a student mark
90
Enter a student mark
79
Enter a student mark
0
85
```

1.4. Nested While Loops

మనం while loop లోపల while loop లను వాడవచ్చు. వీటిని nested while loops అని అంటారు.

Example 12: ఈ ప్రోగ్రాం nested while loops వాడటము ఎలాగో చూపిస్తుంది.

```
#include<stdio.h>
int main()
{
```

```
    int i,j,n;
    printf("Enter a positive Integer\n");
    scanf("%d", &n);
    i=1;
    while(i<=n)
    {
    j=1;
    while(j<=i){
    printf("%d", j);
    j++;
    }
    printf("\n");
    i++;
    }
    return 0;
}
```

Output:

```
1
12
123
```

ఈ కింది పట్టిక ద్వారా ఈ ప్రోగ్రాంను కూడా step-by-step పరిశీలించి చేస్తే మనకు ఇది ఎలా పనిచేస్తుందో అర్థమవుతుంది.

n	i	j	Remarks
3			3ను n ఇద్దామనుకుందాం.
	1		Outer లూప్i kకండిషను ట్రూ, కాబట్టి Outer లూప్ లోనికి కంట్రోలు వెళుతుంది.
		1	inner లూప్i kకండిషను ట్రూ, కాబట్టి inner లూప్ లోనికి కంట్రోలు వెళుతుంది. j value ప్రింట్ చేస్తుంది.
		2	inner లూప్i kకండిషను ఫాల్స్ కాబట్టి inner లూప్i లోనుంచి బయటకు వస్తాం.
	2		i++ అనేది రన్ అవడముతో i విలువ 2 అయ్యి Outer లూప్i kకండిషనుకు వెళుతుంది. Outer లూప్i kకండిషను ట్రూ, కాబట్టి Outer లూప్ లోనికి కంట్రోలు వెళుతుంది.
		1	inner లూప్i kకండిషను ట్రూ, కాబట్టి inner లూప్ లోనికి కంట్రోలు వెళుతుంది. j value ప్రింట్ చేస్తుంది.
		2	inner లూప్i kకండిషను ట్రూ, కాబట్టి inner లూప్ లోనికి కంట్రోలు వెళుతుంది. j value ప్రింట్ చేస్తుంది.
		3	inner లూప్i kకండిషను ఫాల్స్ కాబట్టి inner లూప్i లోనుంచి బయటకు వస్తాం.
	3		i++ అనేది రన్ అవడముతో i విలువ 3 అయ్యి Outer లూప్i kకండిషనుకు వెళుతుంది. Outer లూప్i kకండిషను ట్రూ, కాబట్టి Outer లూప్ లోనికి కంట్రోలు వెళుతుంది.
		1	inner లూప్i kకండిషను ట్రూ, కాబట్టి inner లూప్ లోనికి కంట్రోలు వెళుతుంది. j value ప్రింట్ చేస్తుంది.
		2	inner లూప్i kకండిషను ట్రూ, కాబట్టి inner లూప్ లోనికి కంట్రోలు వెళుతుంది. j value ప్రింట్ చేస్తుంది.
		3	inner లూప్i kకండిషను ట్రూ, కాబట్టి inner లూప్ లోనికి కంట్రోలు వెళుతుంది. j value ప్రింట్ చేస్తుంది.
		4	inner లూప్i kకండిషను ఫాల్స్ కాబట్టి inner లూప్i లోనుంచి బయటకు వస్తాం.
	4		i++ అనేది రన్ అవడముతో i విలువ 4 అయ్యి Outer లూప్i kకండిషనుకు వెళుతుంది. Outer లూప్i kకండిషను ఫాల్స్ కాబట్టి Outer లూప్ లోనుంచి బయటకు వస్తుంది.

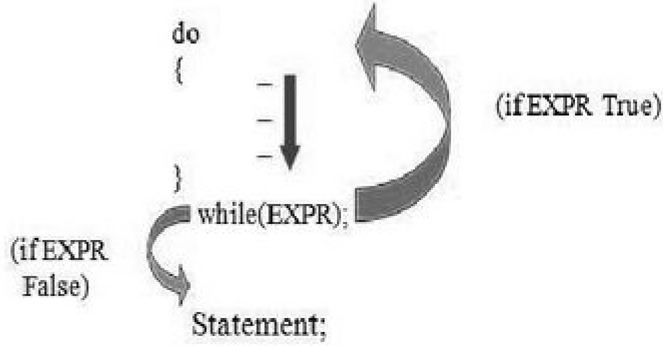
పైన ఇచ్చిన ప్రోగ్రాముని రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter a positive Integer
3
1
12
123
```

2 The do-while Loop

C లాంగ్వేజీలో do-while loop అనే మరొక లూప్ కూడా ఉంది. దీని execution flow ఈ కింద ఇచ్చిన బొమ్మలో చూడవచ్చు. ఇది ముందు బ్లాక్ ను రన్ చేసి ఆ తర్వాత కండిషను (EXPR) చెక్ చేస్తుంది. అది ట్రూ అయితే మరలా బ్లాక్ ను రన్

చేస్తుంది, ఫాల్స్ అయితే లూప్ లోనుంచి బయటకు, అనగా Statement కు వస్తుంది. దీనికీ, ఇంతకు ముందు లూప్ కు ముఖ్యమైన తేడా ఏమిటంటే, ఇది బ్లాక్ ను ఒక్కసారైనా రన్ చేస్తుంది. ఇందులో కూడా మనం break, continue statement లు వాడవచ్చు.



Example 13: ఈ కింది ప్రోగ్రాం do-while loop ఎలా పని చేస్తుందో చూపిస్తుంది.

```
#include<stdio.h>
int main()
{
    int i, n;
    printf("Enter a value for n\n");
    scanf("%d", &n);
    i=0;
    do{
    printf("%d\n", i);
    i++;
    }
    while(i<n);
    return (0);
}
```

ఈ కింది పట్టిక పై ప్రోగ్రాంను trace చేయగా వచ్చింది.

n	i	Output	Remark
5			
	0		Initial value of i. This is printed.
	1	0	i value is incremented
			While condition is tested. Here it is true as i (1) is less than n (5). Thus, control goes to the beginning of do-while block.
		1	i value 1 is printed.
	2		i value is incremented
			While condition is tested. Here it is true as i (2) is less than n (5). Thus, control goes to the beginning of do-while block.
		2	i value 2 is printed.
	3		i value is incremented
			While condition is tested. Here it is true as i (3) is less than n (5). Thus, control goes to the beginning of do-while block.
		3	i value 3 is printed.
	4		i value is incremented
			While condition is tested. Here it is true as i (4) is less than n (5). Thus, control goes to the beginning of do-while block.
		4	i value 4 is printed.
	5		i value is incremented
			While condition is tested. Here it is false as i (5) is not less than n (5). Thus, control goes out of the loop.

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter a value for n
5
0
1
2
3
4
```

Example 14: ఈ కింది ప్రోగ్రాం, 0-100 లోపల ఉన్న సంఖ్య ఇస్తేనే లూప్ లోపలనుంచి బయటకు వచ్చేవిధంగా రాసింది.

```
#include<stdio.h>
int main()
{
    int n;
    do
    {
    printf("Enter a number between 0-100\n");
    scanf("%d", &n);

    }
```

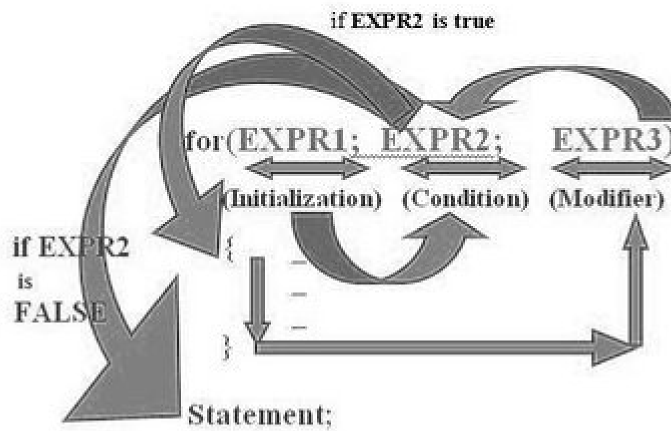
```
    while(n<0 || n>100);
    printf(“Number Entered=%d\n”, n);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter a number between 0-100
190
Enter a number between 0-100
-80
Enter a number between 0-100
75
Number Entered=75
```

3. The FOR Loop

C లాంగ్వేజీలో ఉన్న మరొక ముఖ్యమైన లూప్ ఇది. దీనికీ while లూప్ కు ఏ తేడా లేదు. ఇందులో, initialization statements, condition statements, modifier statements, అన్నింటినీ ఒక దగ్గర, అంటే ఒక లైనులో రాస్తాం. అందుచేత ప్రోగ్రాం సులభంగా అర్ధమవుతుంది. ఈ కింద,లూప్ execution trace చూపించాం. ఇక్కడ కూడా లూప్ బ్లాక్, EXPR2 ట్రూ గా ఉన్నంత వరకు రన్ అవుతుంది, అది ఫాల్స్ అయితే లూప్ లోనుంచి బయటకు అంటే Statement కు వస్తాం.



ఇందులో, EXPR1, EXPR2, EXPR3 లు కామాలతో ఉన్న statement లు లేదా empty statement లు కూడా కావచ్చు. ఈ కింద ఇచ్చినది infinite లూప్ లా పని చేస్తుంది.

```
for( ; ; )
{
}
```

Example 15: ఈ కింది ప్రోగ్రాం for loop ఎలా పని చేస్తుందో చూపిస్తుంది.

```
#include<stdio.h>
```

```
int main()
{
    int i, n;
    printf(Enter an integer\n”);
    scanf(“%d”, &n);
    for(i=0; i<n; i++){
    printf(“%d\n”, i);
    }
    return (0);
}
```

ఈ కింది పట్టిక ద్వారా పై ప్రోగ్రాంను step-by-step ట్రేస్ చేద్దాం.

n	i	రిమార్క్స్
5		n కు 5 విలువ ఇచ్చామనుకొందాము.
	0	i అనే variableకు సున్నా ఇచ్చాం.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది. i value ప్రింట్ చేస్తుంది.
	1	i variableకు 1 కలుపుతాం.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది. i value ప్రింట్ చేస్తుంది.
	2	i variableకు 1 కలుపుతాం.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది. i value ప్రింట్ చేస్తుంది.
	3	i variableకు 1 కలుపుతాం.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది. i value ప్రింట్ చేస్తుంది.
	4	i variableకు 1 కలుపుతాం.
		i విలువ n కంటే తక్కువ కాబట్టి program execution లూప్ లోపలికి వెళుతుంది. i value ప్రింట్ చేస్తుంది.
	5	i variableకు 1 కలుపుతాం.
		i విలువ n సమానం కాబట్టి program execution, లూప్ లోనుంచి బయటకు వెళుతుంది.

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

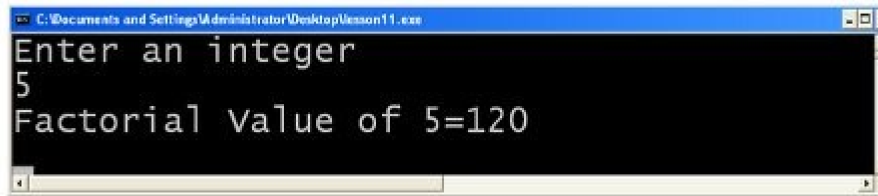


Example 16: ఈ ప్రోగ్రాం ఒక integer (n)ను రీడ్ చేసి దాని factorial విలువను లెక్కిస్తుంది. ఇది అంతకు ముందు చేసినదే. కానీ ఇక్కడ for loop వాడాం.ఇక్కడ ప్రోగ్రాం చిన్నదిగా ఉంది.

```
#include<stdio.h>
int main()
{
    int i, n, F;
    printf("Enter an integer\n");
    scanf("%d", &n);
     for(F=1, i=1; i<=n; i++) F = F*i;

     printf("Factorial Value oof %d=%d\n",n, F);
     return (0);
}
```

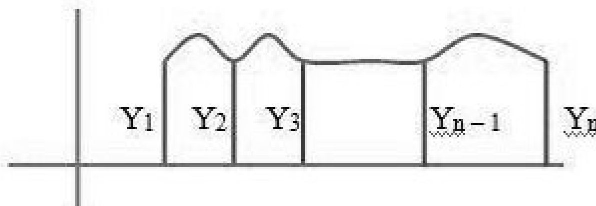
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



Example 17: ఈ ప్రోగ్రాం curve యొక్క height ను 'n' unitly spaced points దగ్గర రీడ్ చేసి curve కింద ఉన్న area ను trapezoidal rule వాడి లెక్కిస్తుంది.

Formula: $(h/2)[Y_1 + Y_n + 2(Y_2 + Y_3 + \ldots\ldots + Y_{n-1})]$

(Here, h value is taken as 1)



Solution: ముందు మొదటి, ఆఖరి points దగ్గర height రీడ్ చేస్తాం. ఆ తర్వాత, n-2 సార్లు రన్ అయ్యే ఒక లూప్ రాసి, ప్రతీసారి ఒక point height రీడ్

చేసి, మొత్తానికి కలుపుతాం. లూప్ లోనుంచి బయటకు వచ్చిన తర్వాత, ఫార్ములా వాడి area ను trapezoidal rule వాడి లెక్కిస్తాం.

```
#include<stdio.h>
int main()
{
    float y1, yn, y, area = 0;
    int i,n;
    printf(“Enter number of points\n”);
    scanf(“%d”, &n);
    printf(“Enter first point height\n”);
    scanf(“%f”,&y1);
    for(i=0;i<n-2;i++)
    {
    printf(“Enter next point height \n”);
    scanf(“%f”, &y);
    area += y;
    }
    printf(“Enter last point height\n”);
    scanf(“%f”, &yn);
    area = 0.5*(y1+yn+(2*area));
    printf(“Area=%f”, area);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of points
7
Enter first point height
98
Enter next point height
78
Enter next point height
34
Enter next point height
44
Enter next point height
44
Enter next point height
341
Enter last point height
412
Area=796.000000
```

Example 18: ఈ ప్రోగ్రాం కూడా curve height ను 'n' unitly spaced points దగ్గర రీడ్ చేసి curve కింద ఉన్న area ను Simpson rule వాడి లెక్కిస్తుంది.

Simpson Formula ప్రకారము

area = (h/3)[Y1 + Yn + 4(Y2 +Y4 + …….) + 2 (Y3 +Y5 + ……. +)]

ఇక్కడ కూడా, ముందు మొదటి, ఆఖరి points దగ్గర height రీడ్ చేస్తాం. ఆ తర్వాత, n-2 సార్లు రన్ అయ్యే ఒక లూప్ రాసి, ప్రతీసారి ఒక point height రీడ్ చేస్తాం. అది సరి index ఉండే point అయితే, దానిని ae కు లేకపోతే ao కు కలుపుతాం. లూప్ లోనుంచి బయటకు వచ్చిన తర్వాత, ఫార్ములా వాడి area ను లెక్కిస్తాం.

```
#include<stdio.h>
int main()
{
    float y1, yn, y, area = 0, ao=0, ae=0;
    int i,n;
    printf("Enter number of points\n");
    scanf("%d", &n);
    printf("Enter first point height\n");
    scanf("%f",&y1);
    for(i=0;i<n-2;i++)
    {
    printf("Enter next point height \n");
    scanf("%f", &y);
    if(i%2) ao +=y ;
    else ae += y;
    }
    printf("Enter last point height\n");
    scanf("%f", &yn);
    area = 0.333333*(y1+yn+2*ao + 4* ae);
    printf("Area=%f", area);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of points
7
Enter first point height
98
Enter next point height
78
Enter next point height
34
Enter next point height
44
Enter next point height
44
Enter next point height
341
Enter last point height
412
Area=839.332520
```

4 Conclusions

ఈ పాఠంలో వివిధ రకాలయిన లూప్ లను ఎలా వాడాలో నేర్చుకున్నాం. ఒక లూప్ ఎలా పని చేస్తుందో తెలుసుకునేందుకు, దానిని పరిశీలించి పట్టిక ఎలా తయారు చెయ్యాలో కూడా నేర్చుకున్నాం.

Arrays

Arrays

ఈ పాఠంలో strings, 1-D arrays, 2-D character arrays, వేరే టైప్ 2-D, 3-D arrays గురించి నేర్చుకుందాం.

13.1. String variables

ఇంతకుముందు నేర్చుకున్న పాఠాల్లో, character variables గురించి తెలుసుకున్నాం. కొన్ని character లు కలిస్తే, string అవుతుంది. ఉదాహరణకు, "rama", "Abhi", "10 Kelso St", మొదలైనవి. అంటే పేర్లు,చిరునామాలు, తదితర సమాచారాన్ని స్టోర్ చేసేందుకు, ప్రాసెస్ చేసేందుకు, string variables అవసరమవుతాయి. string variables ను 1-D arrays అని కూడా అనవచ్చు. Array అంటే ఒకే రకమైన మెంబర్లు (elements) ఉండేది. ఈ కింద ఇచ్చిన పద్ధతుల్లో string variables ను declare చేయవచ్చు.

1. char X[20];

→ Positive integer constant.

ఇక్కడ 'X'ను one dimensional character array లేదా string variable అని అంటారు. String Variabl size ఎప్పుడూ positive integer constant గా మాత్రమే ఉండాలి. అంటే, 20

బదులు వేరేదైనా integer నంబరును మాత్రమే ఇవ్వాలి. పైన ఇచ్చిన declaration వల్ల twenty bytes ఉండే memory ‘X’కు allocate అవుతుంది.

2. #define N 20



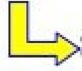
Symbolic constant.

char X[N];

పైన చూపిన విధంగా, String Variable size ను symbolic constant ద్వారా కూడా చెప్పవచ్చు. పైన ఇచ్చిన declaration వల్ల కూడా twenty bytes ఉండే memory ‘X’కు allocate అవుతుంది.

3. అంతే కాకుండా, string variable ను declare చేస్తూ దానికి ఒక string constant ను కూడా కింద చూపిన విధంగా assign చేయవచ్చు.

char X[20] = “RAM ”



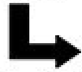
String constant.



పైన ఇచ్చిన declaration వల్ల twenty bytes ఉండే memory ‘X’కు allocate అయి, పై బొమ్మలో చూపినట్లు characters ‘R’, ‘A’, ‘M’ ఒక special character ‘\0’ (null character) ఆ memory లో స్టోర్ అవుతాయి. ఏ string కు అయినా ఈ special character ‘\0’ ఆఖరున add అవుతుంది. దీనిని రాకుండా మనం నియంత్రించలేం. పైన ఇచ్చిన declaration వల్ల ‘X’కు ఇచ్చిన twenty bytes లో మొదటి 4 bytes లో meaningful information (అంటే, ‘R’, ‘A’, ‘M’, ‘\0’) ఉంటుంది, మిగిలిన elements లలో garbage ఉంటుంది.

4. ఈ కింద చూపిన విధంగా కూడా string variable ను declare చేస్తూ దానికి ఒక string constant ను assign చేయవచ్చు. ఇక్కడ string size ను symbolic constant తో చెప్పాం.

#define N 20

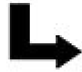


Symbolic constant.

char X[N]=”RAM”;

5. ఈ కింద చూపిన విధంగా కూడా String variable ను దాని size ఇవ్వకుండా declare చేస్తూ ఒక “String Constant” కు assign చేయవచ్చు.

char X[ ] = “RAM ”



String constant.



పైవిధంగా declare చేసిన string కు దానికి assign చేసిన string constant లో ఉన్న characters (null తో కలిపి) స్టోర్ చేసేందుకు ఎంత memory కావాలో అంత memory మాత్రమే allocate అవుతుంది. పైన ఇచ్చిన ఉదాహరణలో 4 bytes మాత్రమే allocate అవుతాయి.

ఈ కింద ఇచ్చిన String variables ను declare చేసేటప్పుడు వాడకూడని పద్ధతులు.

1. char X[-170]; /* Negative size */

2. char X[1.780]; /* Float Array Size */
3. char X["RAM"]; /* size cannot be string */
4. int n = 20; char X[n]; /* size cannot be a variable */
5. char X[ ]; X = "RAM "; /* default size is not acceptable */

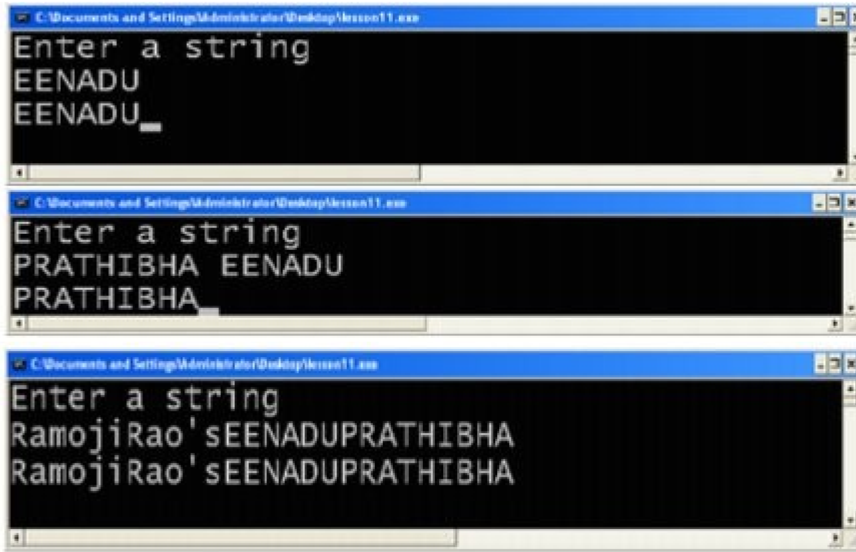
13.1.1 Input/Output string variables using scanf/printf functions

C లాంగ్వేజీలో arrays names ఒక రకంగా memory ను point చేస్తాయి, అంటే memory address ను indicate చేస్తాయి. అందువల్ల, "scanf " ద్వారా "String Variables" లోపలికి input తీసుకునేటప్పుడు address operator(&) string variables కు అప్లై చేయాల్సిన అవసరం లేదు. దీని గురించి pointers chapter లో వివరంగా తెలుసుకుందాం. scanf, printf functions రెండింటిలోనూ string ను రీడ్ లేదా ప్రింట్ చేసేందుకు %s అనేది format string లో వాడుతాం.

ఉదాహరణ 1: ఈ కింది ప్రోగ్రాం scanf(), printf() functions ద్వారా ఒక string ను రీడ్ చేసి ఎలా ప్రింట్ చేయాలో చూపిస్తుంది. ఇక్కడ, %s ను వాడి scanf కు ఒక string ను రీడ్ చేయమని చెబుతున్నాం. అది, space లేదా TAB లేదా \n (newline)వచ్చేంతవరకు characters ను తీసుకుంటుంది. అంటే, మనం "Rama is a good boy" అని keyboard నుంచి ఇస్తే Rama మాత్రమే తీసుకొని scanf() లో ఇచ్చిన string variable లో స్టోర్ చేస్తుంది. అలాగే, "Rama" ఇస్తే Rama ను తీసుకుంటుంది. అంతేకాకుండా ఎన్ని characters తీసుకుంటుంది అనేది string variable size మీద ఆధారపడి ఉంటుంది; keyboard నుంచి string declare చేసిన size - 1 కంటే ఎక్కువ characters ఎప్పుడూ తీసుకోదు.

```
#include<stdio.h>
int main()
{
    char x[20];
    printf("Enter a string\n");
    scanf("%s", x);
    printf("%s", x);
    return (0);
}
```

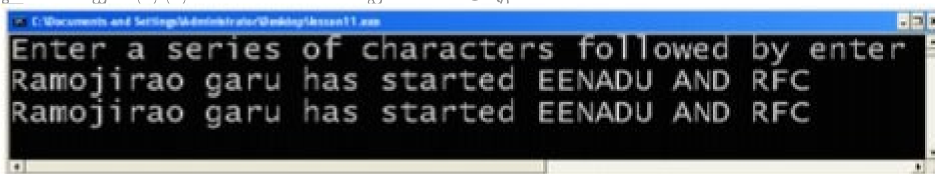
పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



ఉదాహరణ 2: ఈ ప్రోగ్రాం ఒక line full of text ను keyboard నుంచి ఒక string variable లోపలికి తీసుకుని దానినే ప్రింట్ చేస్తుంది. ఇక్కడ scanf() format string లో %[^\n] అనే దాన్ని line full of text ను keyboard నుంచి రీడ్ చేయమని చెప్పేందుకు వాడుతున్నాం. ఇక్కడ, %[^\n] దాని అర్థం, scanf కు newline(enter or return key) ను కీబోర్డు నుంచి user ప్రెస్ చేసేంతవరకు characters ను తీసుకోమని చెబుతున్నట్లు అర్థం. అలాగే, మనం ఒక specified character ఇచ్చేంతవరకు characters ను తీసుకోమని చెప్పడానికి ఇలాంటి format string ను వాడవచ్చు. ఉదాహరణకు, character g ఇచ్చేటంతవరకు characters ను తీసుకోమని చెప్పడానికి format string %[^g]ను వాడతాం. ఇలాంటి format string, printf function లో వాడలేం. కానీ, %s తోనే మనం రీడ్ చేసిన line full of text ను printf తో ప్రింట్ చేయవచ్చు.

```
#include<stdio.h>
int main()
{
    char x[80];
    printf(“Enter a series of characters followed by enter key\n”);
    scanf(“%[^\n]”, x);
    printf(“%s\n”, x);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



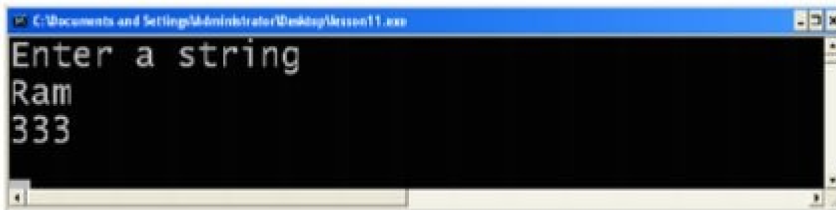
13.1.2 Manipulating Strings As a whole

Strings ను ప్రోసెస్ చేయడానికి చాలా readymade function లు ఉన్నాయి. వాటిని వాడాలంటే ముందుగా string.h ను మన ప్రోగ్రాంలో include చేయాలి. ఉదాహరణకు, strlen అనే function ఒక string తీసుకొని దానిలో ఉన్న characters (null character కాకుండా) ఎన్నో ఇస్తుంది.

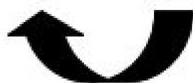
ఉదాహరణ 3: ఈ ప్రోగ్రాం strlen అనే function ను ఎలా వాడాలో తెలుపుతుంది. ఈ function లోపలికి, ఒక string ను పంపాలి, అంటే string variable లేదా string constant ఏదైనా పంపవచ్చని చెప్పడమే ఈ ప్రోగ్రాం ముఖ్య ఉద్దేశం.

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i, j, k;
    char x[80], y[20]= "Ram";
    printf("Enter a string \n");
    scanf("%s", x);
    i = strlen(x);
    j = strlen(y);
    k = strlen("Ram");
    printf("%d%d%d\n", i, j, k);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



Function strcpy: ఒక string లో ఉండే character లను వేరొకదానిలోకి కాపీ చేయడానికి, strcpy ను వాడతాం. దీనికి మనం రెండు string లు ఇస్తే రెండో దానిలో ఉండే characters మొదటి దాని లోపలికి కాపీ చేస్తుంది. దీనిని ఈ కింద చూపిన విధంగా వాడుతాం .



ఉదాహరణ 4: ఈ ప్రోగ్రాం కొన్ని string లను రీడ్ చేసి, వాటిలో పెద్ద string నుప్రింట్ చేస్తుంది.

Solution: మొదటగా ఎన్ని string లు input గా ఇవ్వాలని అనుకుంటున్నారో, అది n అనే variable లోపలికి రీడ్ చేస్తాం. ఆ తర్వాత, n సార్లు రన్ అవుతూ

ప్రతీసారీ ఒక string ను రీడ్ చేసేటట్లు ఒక లూప్ ను రాశాం. తీసుకున్న string length ను max అనే variable తో పోల్చిచూస్తాం. అది max కంటే ఎక్కువ అయితే దానిని max కు ఇచ్చి, ఇచ్చిన string ను ఇంకో string variable(y)లో strcpy function సహాయంతో స్టోర్ (copy) చేస్తాం. లూప్ లోనుంచి బయటకు వచ్చిన తర్వాత, y లో ఉన్న ఫలితం ను ప్రింట్ చేస్తాం.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char x[80],y[80];
    int i, n, m, max = 0;
    printf("Enter number of strings\n");
    scanf("%d", &n);
    i=0;
    while(i<n)
      {
    printf("Enter a string \n");
    scanf("%s", x);
    m = strlen(x);
    if(m>max)
      {
    max = m;
    strcpy(y,x);
           }
    i++;
        }
    printf("\n\nLargest String=%s\n",y);
    return (0);
}
```

ఈ కింద ఇచ్చిన పట్టిక, ఈ ప్రోగ్రాం ఎలా పని చేస్తుందో విశదీకరిస్తుంది.

n	i	x	m	max	y
4	0	Ram		0	
			3		Ram
				3	
	1	Ravin	5		Ravin
				5	
	2	Raja	4		
	3	Rajan	5		

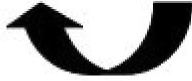
చివరగా Ravin ప్రింట్ అవుతుంది.
పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of strings
5
Enter a string
Ramojirao
Enter a string
EENADU
Enter a string
PRATHIBHA
Enter a string
TELUGU
Enter a string
RFC


Largest String=Ramojirao
```

Function strcmmp: రెండు string లను పోల్చిచూసేందుకు దీనిని ఉపయోగిస్తాం.. దీనికి మనం రెండు string లు ఇస్తే ఇది మనకు ఒక integer ను ఇస్తుంది. మనం పంపిన string లు రెండూ ఒకటే అయితే సున్నా ఇస్తుంది, అలాకాకుండా మొదటిది dictionary order ప్రకారం ముందువస్తే negative సంఖ్య రిజల్ట్ ఇస్తుంది, లేదా positive సంఖ్య ఇస్తుంది. దీనిని కింద చూపిన విధంగా వాడతాం.

strcmp(string1, string2)



ఉదాహరణ 5: ఈ ప్రోగ్రాం కొన్ని string లను రీడ్ చేసి, వాటిలో dictionary order ప్రకారం ముందువచ్చే string ను ప్రింట్ చేస్తుంది.

Solution: దీనిలోకూడా, మొదటగా ఎన్ని string లు input గా ఇవ్వాలనుకుంటున్నారో, అది n అనే variable లోపలికి రీడ్ చేస్తాం. ఆ తర్వాత, n సార్లు రన్ అవుతూ ప్రతీసారీ ఒక string ను రీడ్ చేసేటట్లు ఒక లూప్ ను రాశాం. మొదటసారి (i=0 అయినప్పుడు) తీసుకొన్న string ను y లోపలికి కాపీ చేశాం. ఆ తర్వాత తీసుకొన్న ప్రతీ string ను y తో పోల్చిచూస్తే అది dictionary order ప్రకారం ముందు వస్తే దానిని y లోపలికి strcpy function సహాయంతో స్టోర్(copy) చేస్తాం. లూప్ లోనుంచి బయటకు వచ్చిన తర్వాత, y ను ఉన్న ఫలితం ను ప్రింట్ చేస్తాం..

```
#include<stdio.h>
#include<string.h>
int main()
{
    char x[80], y[80];
```

```
    int i, n;
    printf("Enter number of strings\n");
    scanf("%d", &n);
    for(i=0;i<n;i++)
    {
        printf("Enter a string\n");
        scanf("%s", x);
        if(i==0||(strcmp(y,x)>0)) strcpy(y,x);
    }
    printf("\n\nResult=%s\n", y);
    return (0);
}
```

ఈ కింద ఇచ్చిన పట్టిక, ఈ ప్రోగ్రాం ఎలా పని చేస్తుందో విశదీకరిస్తుంది.

n	i	x	y
5	0	Razaq	
			Razaq
	1	kumar	
	2	Ravindra	
			Ravindra
	3	Abhinav	
			Abhinav
	4	Ravi	

చివరిగా Abhinav ప్రింట్ అవుతుంది.

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of strings
5
Enter a string
Ramojirao
Enter a string
EENADU
Enter a string
PRATHIBHA
Enter a string
TELUGU
Enter a string
RFC


Result=EENADU
```

13.1.3 Manipulating Strings Character by Character

ఇంతకు ముందు చెప్పినట్లుగా, string variable లో character లు elements గా ఉంటాయి. మనం string variables ను character (element) స్థాయుల్లో కూడా ప్రోసెస్ చేయవచ్చు. X అనేది ఒక string variable అయితే X[0], X[1], X[2], మొదలైన వాటిని అందులో ఉండే first,

second, and third characters గా చెప్పవచ్చు. మొదటి element index ఎప్పుడైనా 0 గా ఉంటుంది. దీనిని మనం మార్చలేం.

ఉదాహరణ 6: ఈ కింది ప్రోగ్రాం strlen() function వాడకుండా ఇచ్చిన string length ను ప్రింట్ చేస్తుంది.

Solution: ఒక string రీడ్ చేసి, ఒక్కో character ను null character వచ్చే వరకు ఒక లూప్ వాడి traverse చేస్తాం. ఆఖరున, loop control variable i విలువ string length ను చూపిస్తుంది. దాన్ని ప్రింట్ చేస్తాం.

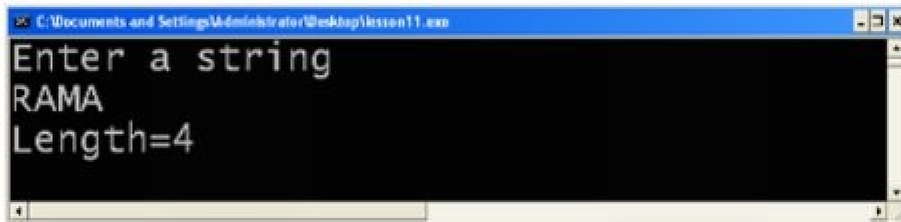
```
#include<stdio.h>
int main()
{
    char x[20];
    int i;
    printf("Enter a string\n");
    scanf("%s", x);
    i=0;
    while(x[i]!='\0')
    {
        i++;
    }
    printf("Length=%d\n", i);
    return (0);
}
```

మనం RAMA ను input గా ఇస్తే పై ప్రోగ్రాంలోని లూప్ ఎలా రన్ అవుతుందో కింద ఇచ్చాం.

0	1	2	3	4															
R	A	M	A	\0															

x[i]	i	x[i]
x[0]	0	'R'
x[1]	1	'A'
x[2]	2	'M'
x[3]	3	'A'
x[4]	4	'\0'

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



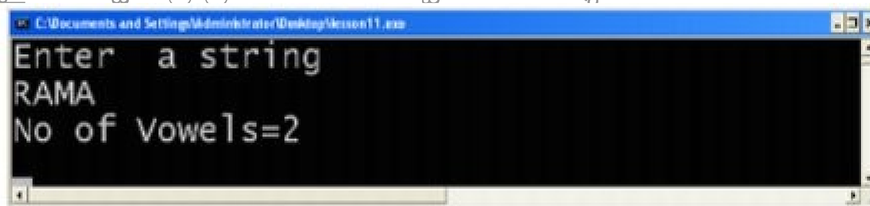
ఉదాహరణ 7: ఈ కింది ప్రోగ్రాం, మనం ఇచ్చిన string లో ఎన్ని "Vowels" ఉన్నాయో ప్రింట్ చేస్తుంది.

Solution: ఒక string రీడ్ చేసి, ఒక్కో character ను null character వచ్చే వరకు ఒక లూప్ వాడి traverse చేస్తాం. ప్రతీ character ను

toupper function వాడి upper case లోనికి convert చేసి switch ద్వారా Vowel అవుతుందేమో చెక్ చేసి, అయితే n విలువను ఒకటి పెంచుతాం. లూప్ లోపలి నుంచి బయటకు వచ్చిన తర్వాత, n విలువను ప్రింట్ చేస్తాం.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    char x[20];
    int i, n = 0;
    printf("Enter a string\n");
    scanf("%s", x);
    i=0;
    while(x[i]!='\0')]
    {
        switch(toupper(x[i])){
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U': n++;
        break;
    }
        i++;
    }
    printf("No of Vowels=%d\n", n);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



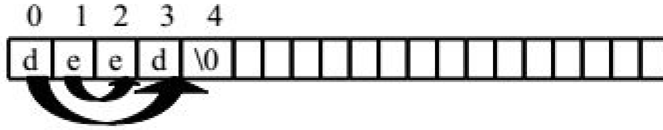
ఉదాహరణ 8: ఈ ప్రోగ్రాం ఒక string ను రీడ్ చేసి అది "Palindrome" అయితే Yes అని లేకుంటే No అని ప్రింట్ చేస్తుంది.

Solution: Palindrome string అంటే దాని reverse దానిలాగే ఉంటుంది. ఉదాహరణలు: "deed", "madam", "malayalam". దీనిని మనం ఇలా చేద్దాం. ఫస్ట్ character ను చివరి character తో పోల్చిచూస్తాం. అవి రెండూ ఒకటే అయితే రెండో character ను చివరి character కు ముందు ఉండే character తో పోల్చిచూస్తాం., అవి రెండూ కూడా ఒకటే అయితే next వాటిని పోల్చిచేస్తాం. ఎప్పుడైనా ఇలా మనం తీసుకొన్న ఏ రెండు character లు అయినా ఒకేలాంటివి కానట్లయితే అది Palindrome కాదని అక్కడే చెబుతాం. String

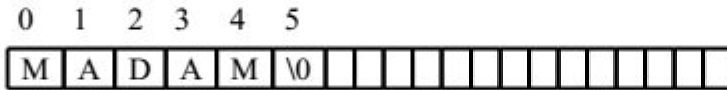
length n అయితే, n/2 character pairs ను పోల్చిచూసుకోవాలి. అన్నీ మ్యాచ్అయితే, string ను Palindrome అని చెబుతాం.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char x[20];
    int i, j, n;
     printf("Enter a string\n");
     scanf("%s", x);
     /*checking whether given string is palindrome or not*/
     n = strlen(x);
     i=0;
     j=n-1;
     while(i<n/2)
      {
     if(x[i]!= x[j]) break;
     i++;
     j--;
     }
     (i==n/2)?printf("Yes\n"): printf("No\n");
     return (0);
}
```

ఉదాహరణకు ఈ కింద ఇచ్చిన ప్రోగ్రాం tracing లను గమనించండి. రెండింటిలోనూ అన్ని (2) character pairs మ్యాచ్అవుతున్నాయి కాబట్టి ఇచ్చింది Palindrome అని వస్తుంది.



i	x[i]	j	x[j]	n	n/2
0	‘d’	3	‘d’	4	2
1	‘e’	2	‘e’		
2					



i	x[i]	j	x[j]	n	n/2
0	‘M’	4	‘M’	5	2
1	‘A’	3	‘A’		
2		2			

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



13.1.4 Processing Line Full of Text

ఉదాహరణ 9: ఈ ప్రోగ్రాం ఒక line full of text ను రీడ్ చేసి దానిలో ఎన్ని పదాలు ఉన్నాయో ప్రింట్ చేస్తుంది.

Solution: ఒక పదంలో దాని చివరి character తర్వాత space అయినా TAB అయినా \0 అయినా ఉండవచ్చు. ఈ logic ను ఉపయోగిస్తూ ఇచ్చిన string ను traverse చేస్తూ ఎన్ని పదాలున్నాయో కనుగొందాం. ఇక్కడ space function ను వాడతాం. ఒక non-space character, దాని పక్కన space character (లేదా null) ఉంటే అది word ending. అప్పుడు ఒక variable, nw, విలువను ఒకటి పెంచుతాం. చివరిలో, nw విలువ ఇచ్చిన లైనులో ఎన్ని పదాలున్నాయో తెలుపుతుంది.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    char x[80];
    int i, nw=0;
    printf("Enter a line full of text \n");
    scanf("%[^\n]", x);
     i=0;
     while(x[i]!='\0')
     {
     if((!isspace(x[i]))&&(isspace(x[i+1])||(x[i+1]=='\0'))) nw++;
     i++;
     }
     printf("No of words=%d\n", nw);
     return (0);
}
```

ఈ కింది పట్టికను పరిశీలిస్తే ఈ ప్రోగ్రాం ఎలా పని చేస్తుందో బాగా అర్థమవుతుంది. ఈ కింద ఇచ్చిన దాన్ని input గా అనుకుందాం.

R	A	M	A				I	S			A		G	O	O	D	\0		

i	x[i]	x[i+1]	nw	Remark
0			0	
	‘R’	‘A’		
1				
	‘A’	‘M’		
2				
	‘M’	‘A’		
3				
	‘A’	‘ ‘		వర్డ్ ending
			1	nw విలువ ఒకటి పెరిగింది.
4	‘ ‘	‘ ‘		
5	‘ ‘	‘ ‘		
6	‘ ‘	‘I’		
7				
	‘I’	‘S’		వర్డ్ ending
			2	nw విలువ ఒకటి పెరిగింది.
8	‘ ‘	‘ ‘		
9	‘ ‘	‘A’		
10	‘A’	‘ ‘		వర్డ్ ending
			3	nw విలువ ఒకటి పెరిగింది.
11	‘ ‘	‘G’		
12	‘G’	‘O’		
13	‘O’	‘O’		
14	‘O’	‘D’		
15	‘D’	‘\0’		వర్డ్ ending
			4	nw విలువ ఒకటి పెరిగింది.
16	‘\0’			లూప్ లోనుంచి బయటకు వస్తాం.

పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter a line full of text
RAMA IS    A      GOOD     BOY
No of words=5
```

13.2. One Dimensional Arrays: Excluding Strings

13.2.1 Why do we need Arrays?

దీనికి సమాధానం తెలియాలి అంటే ముందు ఈ కింది ప్రోగ్రాం గురించి డిస్కస్ చేయాలి. ఇది కొంతమంది విద్యార్థుల మార్కులను రీడ్ చేసి, ఎవరి మార్కులు వారి సరాసరి కన్నా ఎక్కువో అలాంటి వారి సరాసరి ను కాలిక్యులేట్ చేసి ప్రింట్ చేస్తుంది.

```
#include<stdio.h>
int main()
{
    int m, i, s = 0, n1 = 0, s1 = 0, n;
    printf("Enter number of students\n");
    scanf("%d", &n);
    /* reading marks*/
    for(i=0; i<n; i++)
    {
    printf("Enter a student marks\n");
    scanf("%d", &m);
    s = s + m;
    }
    s = s/n; /* class average */
    printf("Enter students marks again\n");
    /*again reading students marks to compare with
average*/
    for(i=0; i<n; i++)
    {
    printf("Enter a student marks\n");
    scanf("%d", &m);
    if(m > s)
    {
    s1 = s1 + m;
    n1++;
    }
    }
    printf("Average =%d\n", s1/n1);
    return (0);
}
```

దీనిలో, మొదటి లూప్ ద్వారా విద్యార్థుల మార్కులను రీడ్ చేసి వారి సరాసరి కనుగొందాం. తర్వాత, రెండో లూప్ ద్వారా మరలా విద్యార్థుల మార్కులను రీడ్ చేసి, ఇంతకు ముందు కనుక్కొన్న సరాసరి తో కంపేర్ చేస్తూ ఎక్కువ ఉన్న వారి మొత్తంమార్కులు, అలాంటి వారు ఎంత మంది ఉన్నారో కనుగొందాం. ఆ తర్వాత, వారి సరాసరి కనుక్కొని ప్రింట్ చేస్తాం.

పైన ఇచ్చిన ప్రోగ్రాముని రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of students
5
Enter a student marks
90
Enter a student marks
80
Enter a student marks
90
Enter a student marks
90
Enter a student marks
80
Enter students marks again
Enter a student marks
90
Enter a student marks
80
Enter a student marks
90
Enter a student marks
90
Enter a student marks
80
Average =90
```

పై ప్రోగ్రాంలో విద్యార్థుల మార్కులను రెండు సార్లు keyboard నుంచి input ఇవ్వాల్సి ఉంటుంది. అలా కాకుండా ఉండాలంటే, మొదటిసారి రీడ్ చేసినప్పుడు ఒక array లో పెడితే వాటిని (విద్యార్థుల మార్కులను) రెండో సారే కాకుండా ఎన్ని సార్లయినా వాడుకోవచ్చు.అప్పుడు మన ప్రోగ్రాం వేగంవతమవుతుంది. అలా, ప్రోగ్రాంలో array లను వాడితే ప్రోగ్రాం ఎఫిషియంట్ అవుతుంది.

ఈ కింద ఇచ్చిన పద్ధతులలో integer array ను declare చేయవచ్చు.

1. int X[20];

→ Positive integer constant.

Array size ఎప్పుడూ positive integer constant గా మాత్రమే ఉండాలి. అనగా, 20 బదులు ఇంకేదైనా integer ను మాత్రమే వాడాలి. పైన ఇచ్చిన declaration వల్ల 20 integers కు కావలసినంత memory 'X'కు allocate అవుతుంది.

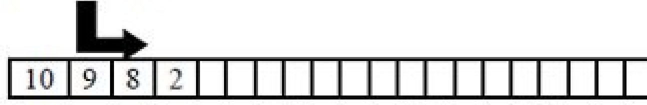
2. #define N 20

→ Symbolic constant.

int X[N];

పైన ఇచ్చిన declaration వల్ల 20 integers కు కావలసినంత memory ‘X’కు allocate అవుతుంది. ఇక్కడ Symbolic constant ను వాడి Array size చెప్పాం.

3. ఈ కింద చూపిన విధంగా కూడా, integer array ను declare చేస్తూ దానికి విలువలును assign చేయవచ్చు.



పైన ఇచ్చిన declaration వల్ల memory ‘X’కు allocate అయి, అందులో పై బొమ్మలో చూపినవిధంగా విలువలు స్టోర్ అవుతాయి; మిగిలిన elements లలో garbage ఉంటుంది.

4. ఇలా కూడా integer array ను declare చేస్తూ దానికి విలువలును ఈ కింద చూపిన విధంగా assign చేయవచ్చు.

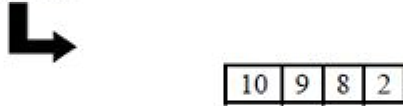
```
#define N 20
```

Symbolic constant.

```
int X[N]= {10,9,8,2};
```

పైన ఇచ్చిన declaration వల్ల కూడా memory ‘X’కు allocate అయి, అందులో పై బొమ్మలో చూపిన విధంగా విలువలు స్టోర్ అవుతాయి; మిగిలిన elements లలో garbage ఉంటుంది.

5. ఈ కింద చూపిన విధంగా integer array ను దాని size ఇవ్వకుండా declare చేస్తూ విలువలును ఈ కింద చూపిన విధంగా assign చేయవచ్చు.



పైన ఇచ్చిన declaration వల్ల ‘X’కు నాలుగు integers కు ఎంత memory కావాలో అంతే allocate అవుతుంది.

ఈ కింద ఇచ్చినవన్నీ integer array ను declare చేసేటప్పుడు వాడకూడని పద్ధతులు.

1) **int a[1.7116]**	Not acceptable as size is float.
2) **int a[-71];**	Not acceptable as size is negative integer.
3) **int N = 10;** **int a[N];**	Not acceptable as size is a variable.
4) **int a[];**	Not acceptable as size is unknown.
5) **int a[10]={1,2,9,,,,12,2};**	Not acceptable as assignment is not valid.
6) **int a[]={1,2,1,,,,22,12};**	Not acceptable as assignment is not valid.
7) **#define N 10** **int a[N]= {1,2,1,,,,22,12};**	Not acceptable as assignment is not valid.
8) **int a[]={1,2,1,,,,22,12};**	Not acceptable as assignment is not valid.
9) **int a["RAM"];**	Not acceptable as size is a string.

పైన చెప్పినవి float, double, long, మొదలగు టైపు arrays కూడా వర్తిస్తాయి.

Example 1: Array ను ఉపయోగించి ప్రోగ్రాం ఎలా రాయాలో పైన చూశాం. ఇందులో విద్యార్థుల మార్కులను ఒక్కసారే రీడ్ చేస్తున్నాం.

```
#include<stdio.h>
int main()
{
    int a[10], i, s = 0, n1 = 0, s1 = 0, n;
    printf("Enter number of students\n");
    scanf("%d", &n);
    /* reading marks*/
    for(i=0; i<n; i++){
    printf("Enter a student mark\n");
    scanf("%d", &a[i]);
    s = s + a[i]; /* Marks are added to s */
    }
    s = s/n; /* Class average is calculated */
    for(i=0; i<n; i++){
    /* Marks are compared with average. If they are more
than class average, the same are added to s1 and n1 value
is incremented by 1 */
    if(a[i] > s) { s1 = s1 + a[i];
    n1++;
    }
    }
    printf("Result=%d\n", s1 / n1);
    /* Averge of those students whose marks are more
than the class average is calculated */
    return (0);
```

}
పైన ఇచ్చిన ప్రోగ్రాం రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of students
10
Enter a student mark
70
Enter a student mark
10
Enter a student mark
81
Enter a student mark
34
Enter a student mark
44
Enter a student mark
48
Enter a student mark
67
Enter a student mark
87
Enter a student mark
99
Enter a student mark
23
Result=80
```

Example 2: ఈ ప్రోగ్రాం ఒక set of integers ను integer array లోకి రీడ్ చేసి రిపీట్ కాని నంబర్లను ప్రింట్ చేస్తుంది.

Solution: ముందు ఒక లూప్ ద్వారా ఓ set of integers ను integer array లోకి రీడ్ చేస్తాం. తర్వాత, Outer లూప్ ద్వారా array లో ఉన్న ఎలిమెంట్స్ ను ఒక్కోటి తీసుకొని inner లూప్ ద్వారా అన్ని ఎలిమెంట్స్ తో కంపేర్ చేస్తాం.

```
#include<stdio.h>
int main()
{
    int i, n, a[10], j, c;
    printf("Enter a number\n");
    scanf("%d", &n);
    for(i=0; i<n; i++) {
    printf("Enter a number\n");
    scanf("%d", &a[i]);
    }
    printf("Unique numbers are:\n");
    for(i=0; i<n; i++) /* To traverse array element by
element */
```

```
    {
    c = 0; /* Counter is initialized to 0 */
    for(j=0; j<n; j++)
    {
    /* comparing ith element with all other elements */
    if(a[i] == a[j]) c++;
    }
    if( c == 1) printf("%d\n", a[i]);
    }
    return (0);
}
```

71	5	71	61	17					

పైన ఇచ్చిన విలువలు array లో ఉన్నాయి .అనుకుంటే, కింద టేబుల్ ద్వారా ట్రేస్ చేశాం.

n	i	a[i]	j	a[j]	c	
5	0	71			0	
			0	71	1	a[i], a[j]లు సమానం కనుక c ఒకటి పెరుగుతుంది.
			1	5		
			2	71	2	a[i], a[j]లు సమానం కనుక c ఒకటి పెరుగుతుంది.
			3	61		
			4	17		
			5			inner loop లో సమానం బయటకు వస్తాం.
						ఇప్పుడు c ఒకటి కాదు, అనగా 71 ఇచ్చిన **array** లో ఒకసారి కన్నా ఎక్కువసార్లు వుంది. కనుక దానిని ప్రింట్ చేయనవసరం లేదు.
	1	5			0	
			0	71		
			1	5	1	a[i], a[j]లు సమానం కనుక c ఒకటి పెరుగుతుంది.
			2	71		
			3	61		
			4	17		
			5			inner loop లో నుంచి బయటకు వస్తాం.
						ఇప్పుడు c ఒకటి, అనగా 5 ఇచ్చిన array లో ఒకసారే ఉంది. కనుక దానిని ప్రింట్ చేస్తాం.

పైన ఇచ్చిన ప్రోగ్రాముని రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter a number
5
Enter a number
71
Enter a number
5
Enter a number
61
Enter a number
71
Enter a number
17
Unique numbers are:
5
61
17
```

Example 3: ఈ ప్రోగ్రాం 'n' students marks, identification నంబర్లను రెండు one dimensional integer arrays లోపలికి రీడ్ చేసి first class వచ్చిన వారి identification నంబర్లను, second class వచ్చిన వారి identification నంబర్లను, third class వచ్చిన వారి identification నంబర్లను ప్రింట్ చేస్తుంది. ఫస్ట్, సెకండ్, థర్డ్ క్లాస్ లకు రావలసిన కనీస మార్కులు 60, 50, 35.

Array "a" Data	71	56	97	31	46	79	8	19	67
Array "ID" Data	111	114	112	118	117	142	156	132	197

ఉదాహరణకు, మార్కులు, identification నంబర్లు, పైన చూపిన విధంగా ఉన్నాయి అనుకుంటే,
మనకు కింద ఇచ్చిన రిజల్ట్స్ రావాలి.

111 112 142 197
114
117

Solution: ముందు ఓ లూప్ తో డాటా రీడ్ చేశాం. తర్వాత మూడు లూప్ లు వాడాం. మొదటి లూప్ లో ప్రతీ విద్యార్థి మార్క్ ను 60 తో కంపేర్ చేసి, ఎక్కువ అయితే వారి identification నంబర్లును ప్రింట్ చేశాం. అలాగే, మరలా ప్రతీ విద్యార్థి మార్క్ ను సెకండ్ క్లాసా కాదా చెక్ చేసి, అయితే వారి identification నంబర్లును ప్రింట్ చేశాం. ఆ తర్వాత మరలా ప్రతీ విద్యార్థి మార్క్ ను థర్డ్ క్లాసా అని చెక్ చేసి, అయితే వారి identification నంబర్లును ప్రింట్ చేశాం.

```
#include<stdio.h>
int main()
```

```
{
    int i, n, a[10], id[10];
    printf("Enter number of students\n");
    scanf("%d", &n);
    for(i=0; i<n; i++){
    printf("Enter marks and IDs of next student\n");
    scanf("%d%d", &a[i], &id[i]);
    }
    /* Prints ID's of first class students */
    printf("First class students IDs\n");
    for(i=0; i<n; i++){
    if(a[i] >= 60) printf("%d ", id[i]);
    }
    /* Prints ID's of second class students */
    printf("\nSecond class students Ids\n");
    for(i=0; i<n; i++){
    if(a[i] >= 50&&a[i] <= 60) printf("%d ", id[i]);
    }
    /* Prints ID's of third class students */
    printf("\nThird class students Ids\n");
    for(i=0; i<n; i++){
    if(a[i] >= 35&&a[i] <= 50) printf("%d ", id[i]);
    }
    return (0);
    }
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of students
9
Enter marks and IDs  of next student
71 111
Enter marks and IDs  of next student
56 114
Enter marks and IDs  of next student
97 112
Enter marks and IDs  of next student
31 118
Enter marks and IDs  of next student
46 117
Enter marks and IDs  of next student
79 142
Enter marks and IDs  of next student
8 156
Enter marks and IDs  of next student
19 132
Enter marks and IDs  of next student
67 197
First class students IDs
111 112 142 197
Second class students Ids
114
Third class students Ids
117
```

Example 4: ఈ ప్రోగ్రాం 'n' students marks, identification నంబర్లను రెండు one dimensional integer arrays లోకి రీడ్ చేసి highest marks వచ్చిన వారి identification నంబర్లను, second highest marks వచ్చిన వారి identification నంబర్లను ప్రింట్ చేస్తుంది.

Array of "a": -	71	56	97	97	46	79	8	79	97	
Array of "ID": -	111	114	112	118	117	142	156	132	197	

ఉదాహరణకు, మార్కులు, identification నంబర్లు, పైన చూపిన విధంగా ఉన్నాయి అనుకుంటే, మనకు కింద ఇచ్చిన రిజల్ట్స్ రావలెను.

112 118 197

142 132

Solution: ముందు ఒక లూప్ తో డాటా రీడ్ చేశాం. తర్వాత లూప్ లో highest marks, second highest marks కనుగొందాం. తర్వాత లూప్ లో ప్రతీ విద్యార్థి మార్కులను highest marks తో కంపేర్ చేసి సమానం అయితే వారి identification నంబర్లను ప్రింట్ చేశాం. అలాగే, మరలా ప్రతీ విద్యార్థి మార్కులను second highest marks తో సమానమా అని చెక్ చేసి, సమానం అయితే వారి identification నంబర్లను ప్రింట్ చేశాం.

```
#include<stdio.h>
int main()
{
    int i, n, a[10], id[10], max1 = 0, max2;
```

```
    printf("Enter number of students\n");
    scanf("%d", &n);
    /* reading marks and identification numbers*/
    for(i=0; i<n; i++){
    printf("Enter marks and IDs of next student\n");
    scanf("%d%d", &a[i], &id[i]);
    }
    /* finding highest and second highest marks*/
    for(i=0; i<n; i++){
    if(a[i] > max1){
    max2 = max1;
    max1 = a[i];
    }
    else if(a[i] > max2 && a[i] < max1){
    max2 = a[i];
    }
    }
    /* Each student mark is compared with max1. If
matches, his ID is printed */
    printf("Ids of students who got highest marks\n");
    for(i=0; i<n; i++) if(a[i] == max1) printf("%d\t",
id[i]);
    printf("\n");
    printf("Ids of students who got second highest
marks\n");
    /* Each student mark is compared with max2. If
matches, his ID is printed */
    for(i=0; i<n; i++) if(a[i] == max2) printf("%d\t",
id[i]);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రామును రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of students
9
Enter marks and IDs of next student
71 111
Enter marks and IDs of next student
56 114
Enter marks and IDs of next student
97 112
Enter marks and IDs of next student
97 118
Enter marks and IDs of next student
46 117
Enter marks and IDs of next student
79 142
Enter marks and IDs of next student
8 156
Enter marks and IDs of next student
79 132
Enter marks and IDs of next student
97 197
Ids of students who got highest marks
112     118     197
Ids of students who got second highest marks
142     132
```

13.2.2 Arrays other than integer 1-D arrays

float, long, double టైప్ 1-D arrays ను కూడా వాడవచ్చు. అన్ని notations వీటితో వాడుకోవచ్చు. ఈ కింది ఉదాహరణలో float array ను ఎలా వాడాలో చూపించాం.

Example 5: ఈ ప్రోగ్రాం ఒక polynomial coefficient ను ఒక float array లోపలికి రీడ్ చేసిన తర్వాత polynomial విలువ, derivative విలువలను మనం ఇచ్చిన point 'x' దగ్గర కాలిక్యులేట్ చేసి ప్రింట్ చేస్తుంది.

```
#include<stdio.h>
#include<math.h>
int main()
{
    int i, n;
    float c[10], x, f, ff;
    printf("The order of the polynomial\n");
    scanf("%d", &n);
    /* nth order polynomial contains n+1 coefficients or
terms */
    for(i=0; i <= n; i++) {
    printf("Enter coefficient of term %d\n", i);
    scanf("%f", &c[i]);
    }
```

```
    printf("Enter a value for x\n");
    scanf("%f", &x);
    f = c[0];
    ff = 0;
    for(i = 1; i <= n; i ++){
    f += c [ i ] * pow( x, i );
    ff += i * c [ i ] * pow( x, i - 1);
    }
    printf("Polynomial Value=%f, Derivative= %f \n", f, ff);
    return (0);
}
```

ఈ కింద polynomial ను ఉదాహరణగా తీసుకొని పై ప్రోగ్రాం ఎలా పని చేస్తుందో చూపించాం. ఇక్కడ x విలువ 2 అని అనుకున్నాం.

Polynomial: $2 - 3x + 4x^2 - 2x^3$
Derivative: $-3 + 8x - 6x^2$

2	-3	4	- 2							

i	x	f	ff
	2	2	0
1		-4 (i.e., 2-3*2)	-3
2		12 (i.e., -4 + 4*2^2)	13(-3+8x2)
3		-4 (i.e., 12 -2*2^3)	-11(13-6x2^2)

పైన ఇచ్చిన ప్రోగ్రాముని రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
The order of the polynomial
3
Enter coefficient of term 0
2
Enter coefficient of term 1
-3
Enter coefficient of term 2
4
Enter coefficient of term 3
-2
Enter a value for x
1.7
Polynomial Value=-1.366001, Derivative= -6.740001
```

13.3. 2-D Character Arrays

C language లో మనం 2-D multi-dimensional arrays ను కూడా వాడవచ్చు. ఇప్పుడు మనం 2-D character arrays గురించి నేర్చుకుందాం..

2D character Arrays ను ఈ కింది విధంగా declare చేయవచ్చు.

I. char X[10][20];

ఇది 10x20 ఉండే 2D character array ను క్రియేట్ చేస్తుంది.

II. char X[5][20] = { “RAM”, “RAVI”, “ABHI”};

ఇది 10x20 ఉండే 2D character array ను క్రియేట్ చేస్తూ మనం assign చేసిన “RAM”, “RAVI”, “ABHI” అనే string constants నుఈకిందివిధంగాస్టోర్చేస్తుంది.

R	A	M	\0																
R	A	V	I	\0															
A	B	H	I	\0															

III. char X[ ][20] = { “RAM”, “RAVI”, “ABHI”};

ఇది 20 columns ఉండే 2D character array ను క్రియేట్ చేస్తూ మనం assign చేసిన “RAM”, “RAVI”, “ABHI” అనే string constants ను ఈ కింది విధంగా స్టోర్ చేస్తుంది. ఇక్కడ rows ఎన్నో మనం ఇవ్వలేదు, మూడు string constants లను assign చేస్తున్నాం కనుక, rows ను మూడుగా తీసుకుంటుంది.

R	A	M	\0																
R	A	V	I	\0															
A	B	H	I	\0															

IV. #define M = 10

#define N = 20

char X[M][N]= { “RAM”, “RAVI”, “ABHI”};

ఇది 10x20 ఉండే 2D character array ను క్రియేట్ చేస్తూ మనం assign చేసిన “RAM”, “RAVI”, “ABHI” అనే string constants ను ఈ కింది విధంగా స్టోర్ చేస్తుంది. ఇక్కడ rows, columns, సైజ్ లను మనం ఇచ్చిన symbolic constants M, N విలువలగా తీసుకుంటుంది.

R	A	M	\0																
R	A	V	I	\0															
A	B	H	I	\0															

X కనుక ఒక two dimensional character array అయితే X[0], X[1], X[2], అనేవి దానిలో ఉన్న rows. వాటినే మనం one dimensional character arrays అని, లేదా string variables అనొచ్చు. అలాగే, X[i][j], ith row jth column element, అది ఒక character. పైన ఇచ్చిన ఉదాహరణలో X[0] అనేది RAM అవుతుంది, X[1] అనేది RAVI అవుతుంది. అలాగే, X[2][2] అనేది ‘H’ అవుతుంది.

Example 1: ఈ కింది ప్రోగ్రాం కొన్ని strings ను two dimensional character array rows లోనికి రీడ్ చేసి వాటిలో longest string ను ప్రింట్ చేస్తుంది.

Solution: ముందుగా ఒక లూప్ రాసి, కొన్ని strings ను two dimensional character array rows లోకి రీడ్ చేస్తాం. తర్వాత, ఇంకో లూప్ సహాయంతో ప్రతి string length ను strlen() function వాడి తెలుసుకుంటూ వాటిలో longest string index ను కనుక్కొంటాం. ఆఖరున, longest string ను ప్రింట్ చేస్తాం.

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i, n, m, id, max = 0;
    char x[10][10];
    printf("Enter number of strings\n");
    scanf("%d", &n);
    /* reading a set of strings into rows of a 2-D character
array*/
    for(i=0; i < n; i++){
    printf("Enter a string\n");
    scanf("%s", x[i]);
    }
    /*finding largest string*/
    for(i=0; i < n; i++){
    m = strlen(x[i]);
    /* length of the string x[i] is calculated */

    if( m > max ) {
    max = m;
    id=i;
    }
    }
    printf("Largest string=%s\n", x[id]);
    return (0);
}
```

R	A	M	\0						
A	B	H	I	N	A	V	\0		
R	A	M	A	\0					
A	N	U	J	A	N	\0			

ఈ ప్రోగ్రాం రన్ అవుతున్నప్పుడు, ఈ పైన ఇచ్చిన strings 2-D array లో ఉన్నాయి అని అనుకొని ప్రోగ్రాంను trace చేసాం.

n	i	x[i]	m	max	Id
4				0	
	0	RAM	3	3	0
	1	ABHINAV	7	7	1
	2	RAMA	4		
	3	ANUJAN	6		

ఇక్కడ, ABHINAV ప్రింట్ అవుతుంది.
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of strings
4
Enter a string
RAM
Enter a string
ABHINAV
Enter a string
RAMU
Enter a string
ANUJAN
Largest string=ABHINAV
```

Example 2: ఈ కింది ప్రోగ్రాం కొన్ని strings ను two dimensional character array rows లోకి రీడ్ చేసి వాటిలో alphabetical order లో ముందు వచ్చే string ను ప్రింట్ చేస్తుంది.

Solution: ఈ ప్రోగ్రాంలో కూడా ముందుగా ఒక లూప్ రాసి, కొన్ని strings ను two dimensional character array rows లోకి రీడ్ చేస్తాం. తర్వాత, ఇంకొక లూప్ సహాయముతో ప్రతి string ను వేరొక string తో strcmp function సహాయంతో కంపేర్ చేసి వాటిలో alphabetical order లో ముందు వచ్చే string index ను కనుక్కొంటాం. ఆఖరున, alphabetical order లో ముందు వచ్చే string ను ప్రింట్ చేస్తాం.

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i, n, id;
    char x[10][20];
    printf("Enter number of strings\n ");
    scanf("%d", &n);
    printf("Enter Strings: ");
    /* reading a set of strings into rows of a 2-D character
array*/
    for(i=0; i < n; i++) {
    printf("Enter a string\n");
    scanf("%s", x[i]);
    }
    id = 0; /* first string is assumed as the first one in order*/
    for(i=1; i < n; i++)
    if( strcmp( x[id], x[i]) > 0 ) id = i;
    printf("String which comes first in alphabetical order=%s\n", x[id]);
    return (0);
}
```

ఈ కింద RAM, RAVI, ABHI, ABHINAV, ANU అనే strings ను ప్రోగ్రాంకు ఇచ్చామనుకొని, trace చేశాం.

n	i	id	x[i]
5		0	RAM
	1		RAVI
	2	2	ABHI
	3		ABHINAV
	4		ANU

అఖరున, x[id], అనగా ABHI ప్రింట్ అవుతుంది.
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of strings
 5
Enter Strings: Enter a string
RAM
Enter a string
RAVI
Enter a string
ABHI
Enter a string
ABHINAV
Enter a string
ANU
String which comes first in alphabetical o
```

Example 3: ఈ కింది ప్రోగ్రాం కొన్ని strings ను two dimensional character array rows లోనికి రీడ్ చేసి వాటిలో "VOWELS" ఎక్కువగా ఉండే string ను ప్రింట్ చేస్తుంది.

Solution: ఈ ప్రోగ్రాంలో కూడా ముందుగా ఒకలూప్ రాసి, కొన్ని strings ను two dimensional character array యొక్క rows లోనికి రీడ్ చేస్తాం. తర్వాత, రెండు nested లూప్ లు వాడి ప్రతీ string లో ఎన్ని "VOWELS" ఉన్నాయో కనుక్కొని వాటిలో "VOWELS" ఎక్కువగా ఉండే string ను కనుక్కొని ప్రింట్ చేస్తుంది.

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i, j, n, max = 0, id, k, l;
    char x[10][20];
    printf("Enter number of strings\n");
    scanf("%d", &n);
    /* reading a set of strings into rows of a 2-D character
array*/
    for(i=0; i < n; i++){
    printf("Enter a string\n");
    scanf("%s", x[i]);
    }
    for(i=0; i < n; i++){l=0;
    for(j=0; x[i][j] != '\0'; j++){
    switch( toupper( x[i][j] )){
```

```
case ‘A’ :
case ‘E’ :
case ‘I’ :
case ‘O’ :
case ‘U’ :
l++;
break;
}
}
if( l > max ){
max = l;
id = i;
}
}
printf(“String having more number of vowels=%s\n”,
x[id]);
return (0);
}
```

R	A	M	A	\0																
A	B	H	I	N	A	V	\0													
R	A	M	\0																	
A	N	U	J	A	N	\0														

ఈ ప్రోగ్రాం రన్ అవుతున్నప్పుడు, ఈ పైన ఇచ్చిన strings 2-D array లో ఉన్నాయి అని అనుకొని ప్రోగ్రాంను trace చేశాం.

i	j	X[i][j]	l	max	id
0			0	0	
	0	R			
	1	A	1		
	2	M			
	3	A	2		
	4	\0			
				2	0
1			0		
	0	A	1		
	1	B			
	2	H			
	3	I	2		
	4	N			
	5	A	3		
	V	V			
				3	1
2			0		

అఖరున, x[id], అనగా "VOWELS" ఎక్కువగా ఉండే ప్రింట్ అవుతుంది..
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

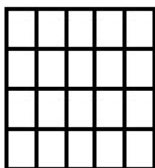
```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of strings
4
Enter a string
RAMA
Enter a string
ABHINAV
Enter a string
RAM
Enter a string
ANUJAN
String having more number of vowels=ABHINA
```

12. 2-D and multi-dimensional integer, float, long, double arrays

ఈ కింద 2-D integer arrays ఎలా declare చేయాలో చూపించాం.

I. int a[4][5];

ఇది 4 rows, 5 columns ఉండే 2-D integer array ను declare చేస్తుంది.



II. int a[4][5] = {{ 1, 7, 1}, { 7, 2, 1}, { 3, 1, 3}};
ఇది 4 rows, 5 columns ఉండే 2-D integer array ను declare చేస్తూ మనం assign చేసిన విలువలను కింది విధంగా స్టోర్ చేస్తుంది.

1	7	1		
7	2	1		
3	1	1		

III. int a[][5] = {{ 1, 7, 1}, { 7, 2, 1}, { 3, 1, 3}};
ఇది 5 columns ఉండే 2-D integer array ను declare చేస్తూ మనం assign చేసిన విలువలను ఈ కింది విధంగా స్టోర్ చేస్తుంది. ఇక్కడా rows ఎన్నో మనం ఇవ్వలేదు. కాని assignment statement నుంచి మూడు rows గా కంపైలరు అనుకుంటుంది.

1	7	1		
7	2	1		
3	1	1		

IV. int a[4][5]={1,2,2,2,2,2,1,1,1};
ఇది 4 rows, 5 columns ఉండే 2-D integer array ను declare చేస్తూ మనం assign చేసిన విలువలను కింది విధంగా స్టోర్ చేస్తుంది.

1	2	2	2	2
2	1	1	1	

V. int a[][5]={1,2,2,2,2,2,1,1,1};
ఇది 4 rows, 5 columns ఉండే 2-D integer array ను declare చేస్తూ మనం assign చేసిన విలువలను కింది విధంగా స్టోర్ చేస్తుంది.

1	2	2	2	2
2	1	1	1	

VI. int a[2][4][5];
ఇది ఒక 3-D integer array ను దానిలో 2 plane లు 4x5 size ఉండేటట్లు declare చేస్తుంది.

Plane 0:

Plane 1:

VII. int a[2][4][5]={ {{1,2}, {1,4}}, {{3,4},{4,5}}};
ఇది 4 rows, 5 columns ఉండే 2-D integer array ను declare చేస్తూ మనం assign చేసిన విలువలను కింది విధంగా స్టోర్ చేస్తుంది.

Plane 0:

1	2			
1	4			

Plane 1:

3	4			
4	5			

VIII. int a[][4][5]={ {{1,2}, {1,4}}, {{3,4},{4,5}}};
ఇది 4 rows, 5 columns ఉండే 2-Dinteger array ను declare చేస్తూ మనం assign చేసిన విలువలను కింది విధంగా స్టోర్ చేస్తుంది.

Plane 0:

1	2			
1	4			

Plane 1:

3	4			
4	5			

IX. int a[2][4]
[5]={1,2,3,4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,6,6,6,6,6,9,4,5
};
ఇది 4 rows, 5 columns ఉండే 2-D integer array ను declare చేస్తూ మనం assign చేసిన విలువలను కింది విధంగా స్టోర్ చేస్తుంది.

Plane 0:

1	2	3	4	4
4	4	4	4	4
4	4	4	5	5
5	5	5	5	6

Plane 1:

6	6	6	6	9
4	5			

X. int a[][4]
[5]={1,2,3,4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,6,6,6,6,6,9,4,5
};
ఇది 4 rows, 5 columns ఉండే 2-D integer array ను declare చేస్తూ మనం assign చేసిన విలువలను కింది విధంగా స్టోర్ చేస్తుంది. ఇక్కడ data నుంచి two planes ఉంటాయని క 0 పైలరు అనుకొంటుంది.

Plane 0:

1	2	3	4	4
4	4	4	4	4
4	4	4	5	5
5	5	5	5	6

Plane 1:

6	6	6	6	9
4	5			

Example 4: ఈ ప్రోగ్రాం 2-D integer array లోపలికి డాటా ఎలా element తర్వాత element ఎలా రీడ్ చేయాలి, ఎలా ప్రింట్ చేయాలో చూపిస్తుంది.

```
#include<stdio.h>
int main()
```

```
{
    int i, j, m, n, a[4][5];
    printf("Enter number of rows and columns\n");
    scanf("%d %d", &m, &n);
    /* Reading a 2-D Array */
    for(i=0; i < m; i++){
    for(j=0; j < n; j++) {
    printf("Enter next element value\n");
    scanf("%d", &a[i][j]);
    }
    }
    /* Printing a 2-D Array */
    printf("Given matrix\n");
    for(i=0; i < m; i++){
    for(j=0; j < n; j++) printf("%d\t", a[i][j]);
    printf("\n");
    }
    return (0);
}
```

ఈ కింది టేబుల్ 2x3 size 2-D array లోపలికి డాటా ఎలా రీడ్ చేస్తుందో చూపిస్తుంది.

m	n	i	j	a[i][j]	Remarks
2	3				
		0	0	1	1 input ఇస్తే అది a[0][0]లో స్టోర్ అవుతుంది.
			1	7	7 input ఇస్తే అది a[0][1] లో స్టోర్ అవుతుంది.
			2	1	1 input ఇస్తే అది a[0][2] లో స్టోర్ అవుతుంది.
			3		inner for loopలో నుంచి బయటకు వస్తాం. ఈ లోపల 1^{st} row లోపలికి డాటా వచ్చి ఉంటుంది.
		1	0	7	7 input ఇస్తే అది a[1][0] లో స్టోర్ అవుతుంది.
			1	2	2 input ఇస్తే అది a[1][1] లో స్టోర్ అవుతుంది.
			2	1	1 input ఇస్తే అది a[1][2] లో స్టోర్ అవుతుంది.
			3		inner for loop లో నుంచి బయటకు వస్తాం. ఈ లోపల 2^{nd} row లోపలికి డాటా వచ్చి ఉంటుంది.
		2			outer for loop లో నుంచి బయటకు వస్తాం. ఈ లోపల అన్ని elements డాటా వచ్చి ఉంటుంది.

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter number of rows and columns
2 3
Enter next element value
1
Enter next element value
7
Enter next element value
1
Enter next element value
7
Enter next element value
2
Enter next element value
1
Given matrix
1       7       1
7       2       1
```

Example 5: ఈ కింది ప్రోగ్రాం ఒక square matrix లోపలికి డాటా రీడ్ చేసి దాని “Principal Diagonal” elements యొక్క టోటల్ ప్రింట్ చేస్తుంది.

Solution: పైన చెప్పిన విధంగా square matrix లోపలికి డాటా రీడ్ చేస్తాం. Principal Diagonal elements అనగా a[0]0], a[1][1], a[2][2], మొదలైనవి. కనుక ఒక లూప్ రాసి వాటి టోటల్ కనుక్కుంటాం.

```
#include<stdio.h>
int main()
{
    int i, j, n, s1 = 0, a[20][10];
    printf(”Enter square matrix size\n”);
    scanf(“%d”, &n);

    /* Reading the Data*/
    printf(”Enter Matrix Data\n”);
    for(i=0; i < n; i++)
    for(j=0; j < n; j++) {
    printf(”Enter element value\n”);
    scanf(“%d”, &a[i][j]);
    }
    /* Calculating Diagonal Sums */
    for(i=0; i < n; i++){
    s1 += a[i][i];
    }

    printf(“Principal Diag. Sum=%d \n”, s1);
```

```
    return (0);
}
```

ఈ కింద డాటా అనుకొని, ప్రోగ్రాంను trace చేస్తాం.

1	7	1		
7	2	1		
3	1	1		

n	i	a[i][i]
3	0	1
	1	2
	2	1

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter square matrix size
3
Enter Matrix Data
Enter element value
1
Enter element value
7
Enter element value
1
Enter element value
7
Enter element value
2
Enter element value
1
Enter element value
3
Enter element value
1
Enter element value
1
Principal Diag. Sum=4
```

Example 6: ఈ ప్రోగ్రాం ఒక square matrix లోపలికి డాటా రీడ్ చేసి అది symmetric matrix అవునా? కాదా? అని ప్రింట్ చేస్తుంది.

Solution: ఒక matrix ను symmetric అనాలంటే దాని transpose దానిలాగే ఉండాలి. దీనినే మరొక విధంగా కూడా చెప్పవచ్చు. దాని lower triangular portion లో ఉన్న elements అన్నీ upper triangular portion లోఉన్న elements తో సమానంగా ఉంటే అది symmetric matrix అవుతుంది. అంటే, elements

a[1][0],
a[2][0], a[2][1]
a[3][0],a[3][1],a[3][2]

ఈ కింది elements లాగా ఉండాలి.

a[0][1], a[0][2], a[0][3]
a[1][2], a[1][3],
a[2][3].

```
#include<stdio.h>
int main()
{
    int i, j, n, a[20][10];
    printf(”Enter square matrix size\n”);
    scanf(“%d”, &n);
    /* Reading the Data*/
    for(i=0; i < n; i++)
    for(j=0; j < n; j++) {
    printf(”Enter element value\n”);
    scanf(“%d”, &a[i][j]);
    }
    for(i=1; i<n;i++)
    for(j=0; j<i;j++){
    if( a[i][j] !=a[i][n – 1 – i ]){
    printf(“No\n”);
    exit(-1);
    }
    }
    printf(“Yes\n”);
    return (0);
}
```

ఈ కింది డాటా అనుకొని, పై ప్రోగ్రాంను trace చేశాం.

1	7	1	8	
7	2	1	9	
1	1	1	1	
4	5	3	3	

n	i	j	a[i][j]	a[i][n-1-j]	Remarks
4					
	1	0	7	7	Pairs రెండూ సమానం.
		1			j లూప్ లో నుంచి బయటకు వస్తాం.
	2	0	1	1	Pairs రెండూ సమానం.
		1	1	1	Pairs రెండూ సమానం.
		2			j లూప్ లో నుంచి బయటకు వస్తాం..
	3	0	4	8	Pairs రెండూ సమానం కాదు. అందువలన, matrix symmetric కాదు.

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
1
Enter element value
8
Enter element value
7
Enter element value
2
Enter element value
1
Enter element value
9
Enter element value
1
Enter element value
1
Enter element value
1
Enter element value
1
Enter element value
4
Enter element value
5
Enter element value
3
Enter element value
3
No
```

Example 7: ఈ ప్రోగ్రాం ఒక square matrix లోపలికి డాటా రీడ్ చేసి దాని transpose ను అదే memory లో స్టోర్ చేస్తుంది. ఈ విధంగా స్టోర్ చేయడాన్ని In-situ, In-Core, In-Memory, In-Place operations అని అంటారు.

Solution: ఇది పై ప్రోగ్రాంకు చాలా దగ్గరగా ఉంటుంది. ఇక్కడ elements ను

exchange చేస్తాం. అంటే,
a[1][0],
a[2][0], a[2][1]
a[3][0],a[3][1],a[3][2]
లను ఈ కింది వాటితో exchange చేస్తాం.
a[0][1], a[0][2], a[0][3]
a[1][2], a[1][3],
a[2][3].

```
#include<stdio.h>
int main()
{
    int i, j, n, T, a[20][10];
    printf(”Enter square matrix size\n”);
    scanf(“%d”, &n);
    /* Reading the Data*/
    printf("Enter matrix data\n");
    for(i=0; i < n; i++)
    for(j=0; j < n; j++) {
    scanf(“%d”, &a[i][j]);
    }
    /* Exchanging*/
    for(i=1; i<n;i++)
    for(j=0; j<i;j++){
    T=a[i][j];
    a[i][j]=a[i][n-1-i];
    a[i][n-1-i]=T;
    }
    /* Printing the transpose matrix*/
    printf("Transpose\n");
    for(i=0; i < n; i++){
    for(j=0; j < n; j++) {
    printf(“%d\t”, a[i][j]);
    }
    printf(”\n”);
    }
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter square matrix size
4
Enter matrix data
1 7 1 8
7 2 1 9
1 1 1 1
4 5 3 3
Transpose
1       7       1       8
1       2       7       9
1       1       1       1
3       4       5       3
```

Example 8: ఈ కింద ఇచ్చిన ప్రోగ్రాం రెండు matrix లను multiply చేసి resultant matrix ను కాలిక్యులేట్ చేస్తుంది.

Solution: రెండు matrix లను multiply చేయాలంటే మొదటి matrix cols రెండో matrix rows తో సమానంగా ఉండాలి. m మొదటి matrix ప్రతీ row ను రెండో matrix ప్రతీ column తో multiply చేస్తాం.

```
#include<stdio.h>
int main()
{
    int i, j, k, m1, n1, m2, n2, A[20][20], B[20][20], C[20]
[20];

    printf("Enter the rows and columns of first matrix\n");
    scanf("%d%d", &m1, &n1);
    printf("Enter the rows and columns of second
matrix\n");
    scanf("%d%d", &m2, &n2);
    if(n1!=m2){
    printf("Matrix multiplication is not possible\n");
    exit(-1);
    }
    /* Reading first matrix*/
    printf("Enter Data for first matrix\n");
    for(i=0; i < m1; i++)
    for(j=0; j < n1; j++)
    scanf("%d", &A[i][j]);
```

```
    /* Reading first matrix*/
    printf("Enter Data for second matrix\n");
    for(i=0; i < m2; i++)
    for(j=0; j < n2; j++)
    scanf("%d", &B[i][j]);
    /* Calculating the product matrix*/
    for(i=0; i < m1; i++)
    for(j=0; j < n2; j++){
    C[i][j] = 0;
    for(k=0; k < n1; k++)
    C[i][j] += A[i][k] * B[k][j];
    }
    printf("Resultant Matrix\n");
    for(i=0; i < m1; i++){
    for(j=0; j < n2; j++)
    printf("%d\t", C[i][j]);
    printf("\n");
    }
    return (0);
}
```

ఈ కింద ఇచ్చిన matrix లు డాటా అనుకొని, పై ప్రోగ్రాంను trace చేస్తాం.

2	1	3
7	1	2

1	2
2	1
1	2

7	11
11	19

m1	n1	m2	n2	i	j	k	a[i][k]	b[k][j]	c[i][j]
2	3	3	2	0	0				0
						0	2	1	2
						1	1	2	4
						2	3	1	**7**
					1				0
						0	2	2	4
						1	1	1	5
						2	3	2	**11**
				1	0				0
						0	7	1	7
						1	1	2	9
						2	2	1	**11**
					1				0
						0	7	2	14
						1	1	1	15
						2	2	2	**19**

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
Enter the rows and columns of first matrix
2 3
Enter the rows and columns of second matrix
3 2
Enter Data for first matrix
2 1 3
7 1 2
Enter Data for second matrix
1 2
2 1
1 2
Resultant Matrix
7       11
11      19
```

13.3 Conclusions

ఈ పాఠంలో strings, 1-D arrays, 2-D character arrays, 2-D arrays గురించి నేర్చుకున్నాం.

FUNCTIONS

14.1. Introduction

ఇప్పటి వరకు మనం scanf(), printf(), sqrt(), pow(), strlen(), strcmp(), strcpy(), మొదలైన readymade functions ను ఉపయోగించి ప్రోగ్రాంలు రాశాం. అంటే ఎవరో ముందుగా రాసిన ఈ function ల కోడ్ ను వినియోగించుకుంటున్నారు. అంటే, వాటి కోడ్ కు re-usability పెరిగింది. ఇవి అందుబాటులో లేనట్లయితే మనం ఈ ప్రోగ్రాంలు రాయాల్సి వచ్చేది. అంటే, వీటి ని ఉపయోగించడం ద్వారా మన ప్రోగ్రాం డెవలప్ మెంట్ సులభతరం అయింది. అంతే కాకుండా ఈ function ల కోడ్ ను వాడిన ప్రతీ ప్రోగ్రాంలో test చేయాల్సిన అవసరం లేదు.

ఈ విధంగా, Functions వల్ల కింది ఉపయోగాలు పొందవచ్చు.

1. ప్రోగ్రాం డెవలప్ మెంట్ సులభతరం అవుతుంది.
2. ప్రోగ్రాం డెవలప్ మెంట్ modular అవుతుంది.
3. ప్రోగ్రాం testing సులభతరం అవుతుంది.
4. Code ను share చేసుకోవచ్చు.
5. Code re-usability పెరుగుతుంది
6. ప్రోగ్రాం size తగ్గుతుంది.
7. ప్రోగ్రాం readability పెరుగుతుంది.

ఇప్పుడు, మనం కొత్త function లను రాసే విధానాన్ని ఈ పాఠంలో నేర్చుకుందాం. ప్రతీ function కు ఒక రెస్పాన్సిబిలిటీ ఉంటుంది. అంటే అది కొన్ని argument లను తీసుకుని ఒక రకమైన విలువను ఇస్తుంది. (return చేస్తుంది). ఉదాహరణకు, strlen function ఒక string ను argument గా తీసుకొని, దానిలో ఎన్ని characters ఉన్నాయో ఇస్తుంది. ఏదైనా Function రాస్తున్నపుడు, ఈ కింది విషయాలు గుర్తుంచుకోవాలి.

1. Function పేరు unique గా ఉండాలి. అది C లాంగ్వేజి Key-words ఉండకూడదు, మరింకే function పేరులా కూడా ఉండకూడదు.
2. అది ఎన్ని, ఏ రకమైన argument లను తీసుకుంటుంది .
3. ఏ రకమైన విలువ ఇవ్వాలి.

ఒక function definition ఈ విధంగా ఉంటుంది.

```
return_type function_name( type arg1, type arg2, …., type argn)
    {
    --
    }
```

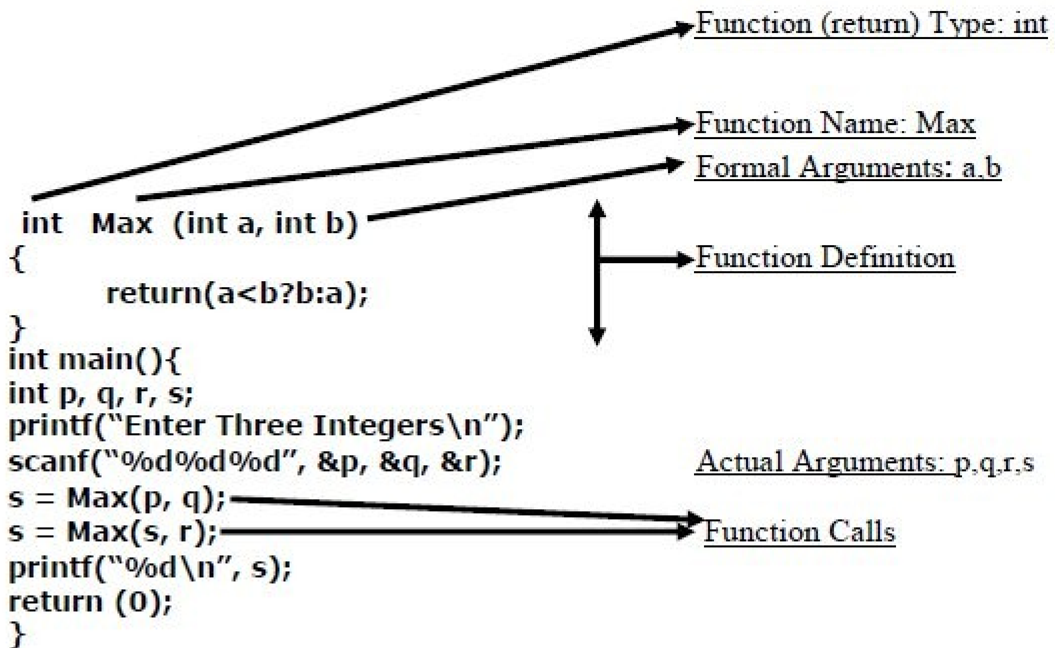
(లేదా)

```
return_type function_name( arg1, arg2,arg3 …..,argn)
    type agr1, arg2;
    type arg3, agrn;
    {
    --
    }
```

Example 1: రెండు పూర్ణాంకాలు(integers) తీసుకొని వాటిలో maximum return చేసే ఒక function రాద్దాం. ఒక main program దానిని call చెయ్యడం ఎలాగో చూపించడానికి రాశాం.

Solution:1. దీని పేరును Max అని అనుకుందాం (వేరే ఇంకేదైనా కూడా తీసుకోవచ్చు).
2. దీనికి రెండు arguments కావాలి కనుక వాటిని, int a, int b అని అనుకుందాం (వేరే ఇంకేవైనా కూడా తీసుకోవచ్చు).
3. మనం రాయాల్సిన function, ఇచ్చిన రెండు integers లో maximum return చెయ్యాలంటే ఈ function int ను return చేయాలి. ఎందుకంటే, రెండు integers లో maximum integer అవుతుంది.

పై points ను , function రాసి దానిని ఎలా call చెయ్యాలో కూడా ఇచ్చాం.



పైన ఇచ్చిన ప్రోగ్రాం main() నుంచి మనం రాసిన Max() function రెండు సార్లు call చేస్తున్నాం. మొదటిసారి, p, q లను actual arguments గా పంపుతాం.వీటి maximum మన Max() function రిటర్న్ చేస్తే దానిని s లో పెట్టాం. అలాగే,

Max() function ను రెండో సారి call చేస్తూ s, r లను పంపించాం. వచ్చిన దానిని s లో స్టోర్ చేసి ప్రింట్ చేసాం. అవసరమైతే ఈ రెండు function call లను s=Max(Max(p,q),r) గా కూడా రాయవచ్చు.

Function definition కాబట్టి Function call తర్వాత ఉంటే, function prototype (function declaration లేదా function signature), function call కన్నా ముందు ఉండేటట్లు చూసుకోవాలి. Function p. Prototype ఈ కింది విధంగా ఉంటుంది. Function prototype అంటే ఒక విధంగా Function definition లో మొదటి లైను అనుకోవచ్చు. కానీ దీనిలో చివరలో సెమీకోలన్ (;)వాడాలి.

(Note: Header file లలో readymade functions కి సంబంధించినా prototypes ఉంటాయి). Function prototype ఇలా ఉంటుంది.)

return_type function_name(type arg1, type arg2, ….,type argn);.

Example 2: ఈ ప్రోగ్రాం, function prototype ను ఎలా వాడాలో చూపిస్తుంది. , ఒక main program దానిని call చేయడం ఎలాగో చూపించడానికి రాశాం.

Solution:

```
#include<stdio.h>
int main()
{
    int p, q, r, s;
    int Max(int xxx, int y);
```



Function Prototype

```
    printf("Enter Three Integers\n");
    scanf("%d %d %d", &p, &q, &r);
    s = Max(p, q);
    s = Max(s, r);
    printf("Max of Three Numbers=%d\n", s);
    return (0);
    }
    int Max(int a, int b)
     {
    return(a<b?b:a);
}
```

అవసరమైతే function definition ను ఒక separate file లో స్టోర్ చేసి దానిని ఏ ప్రోగ్రాంలో

కావాలి అంటే అందులో వాడుకోవచ్చు. అలా వాడుకోవాలంటే, మనం ఆ file ను మన ప్రోగ్రాంలో include చేయాలి.

Example 3: ఈ ప్రోగ్రాం, మన function code file "a.c" అనుకుంటుంది. ఒక main program దానిని call చేయడం ఎలాగో చూపించడానికి రాశాం.

Solution:

```
#include<stdio.h>
#include "a.c"
int main()
{
    int p, q, r, s;
    printf("Enter Three Integers\n");
    scanf("%d %d %d", &p, &q, &r);
    s = Max(Max(p, q), r);
    printf("%d\n", s);
    return (0);
}
```

14.1.2. void టైప్ అంటే ఏమిటి?

ఏదైనా function ఏమీ రిటర్న్ చేయకపోతే దానిని void టైప్ అని అంటారు.
ఉదాహరణకు:
void xyz(void) { }.

Example 1: ఈ ప్రోగ్రాంలో ఒక function ను రాసి దానిని main నుంచి call చేస్తున్నాం. రాసిన function ఒక integer(పూర్ణాంకం) తీసుకొని దాని అంకెల మొత్తాన్ని రిటర్న్ చేస్తుంది. ఒక main program దానిని call చెయ్యడం ఎలాగో చూపించడానికి రాశాం.

```
#include<stdio.h>
#include<stdlib.h>
int sum_of_digits(int n)
{
    int sum = 0;
    while (n > 0)
    {
    sum += n % 10;
    n = n/10;
    }
return sum;
}
int main()
    {
    int n, s;
    printf("Please enter a non-negative integer number\n");
    scanf("%d", &n);
    if(n < 0){
    printf("The number must be non-negative!\n");
    exit( -1);
    }
    s = sum_of_digits(n);
    printf("The sum of digits of %d is %d\n", n, s);
    return 0;
}
```

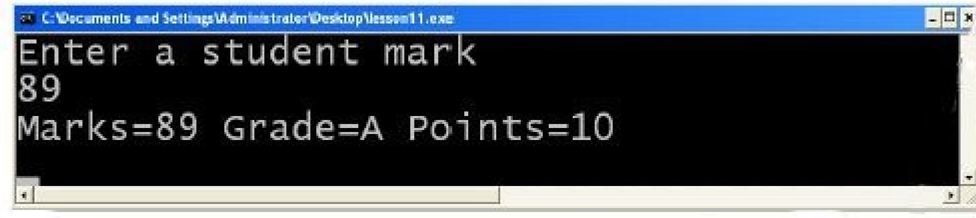
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Please enter a non-negative integer number
1789
The sum of digits of 1789 is 25
```

Example 2: ఒక function ఒక విద్యార్థి మార్కులను తీసుకుని గ్రేడ్ రిటర్న్ చేయాలి. అలాగే, వేరొక function విద్యార్థి గ్రేడ్ తీసుకొని గ్రేడ్ పాయింట్లను రిటర్న్ చేయాలి. main() ప్రోగ్రాం పై రెండింటిని call చేయాలి.
ఎన్ని మార్కులు వస్తే ఏ గ్రేడ్ ఇవ్వాలి? అనేది character variables అనే పాఠంలో రాసిందే ఇక్కడ కూడా వాడుతున్నాం.

```
#include<stdio.h>
char Grade(int n) /* Takes an integer and returns a
character */
    {
    return(‘A’ + (4 - n/20) +( n ==100) );
    }
int Points(char g) /* Takes a character and returns integer
*/
    {
    return(2 * ( 70-g) );
    }
int main()
{
    int m, p;
    char v;
    printf(“Enter a student mark\n”);
    scanf(“%d”, &m);
    v = Grade(m);/* function call*/
    p = Points(v); /* function call*/
    printf(“Marks=%d Grade=%c Points=%d\n”, m, v, p);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



Example 3: ఏదైనా function ఒక character తీసుకొని అది ‘Vowel’ అయితే 1 లేకుంటే 0 రిటర్న్ చేయాలి. ఇక్కడ రెండు విధాలుగా రాశాం. అలాగే main program లో ఒక string ను రీడ్ చేసి దానిలో ఎన్ని ‘Vowels’ ఉన్నాయో పై function తో కనుక్కొని ప్రింట్ చేస్తాం.

```
#include<stdio.h>
#include<ctype.h>
/* Takes a character and returns 1 if it is a vowel. Else
returns 0 */
int Isvowel(char V)
    {
    V = toupper(V);
    return((V==‘A’) || (V==‘E’) || (V==‘I’) || (V==‘O’) ||
(V==‘U’));
    }
(లేదా)
int I is vowel(char V)
     {
    V = toupper(V);
    switch(V)
     {
    case ‘A’ :
    case ‘E’ :
    case ‘I’ :
    case ‘O’ :
    case ‘U’ : return(1);
    default : return(0);
    }
    }
int main()
{
    int i, n = 0;
```

```
    char x[20];
    printf("Enter a string\n");
    scanf("%s", x);
    for(i = 0; x[i] != '\0'; i++)
    if(I is vowel(x[i])) n++;
    /* Here, we are x[i], ith character of the string x to
check
    whether it is vowel or not. If vowel, n value is
    incremented. Thus, n value is number of vowels. */

    printf("No of Vowels in the given string=%d\n", n);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



14.1.3. Passing 1-D Arrays to functions

ఒక function లోకి ఒక 1-D array argument అని చెప్పడానికి, ఈ కింది విధంగా empty brackets పెడతాం.

return_type function_name (Type arg[ ], ...)

Example 1: ఏదైనా function ఒక string ను తీసుకుని దాని length రిటర్న్ చేయాలి. ఒక main program దానిని call చెయ్యడం ఎలాగో చూద్దాం.

```
#include<stdio.h>
    int Length(char x[] ) /* Here, x is a string variable*/
{
    int i = 0;
    while(x[i] != '\0') i++;
    /* We know after this while loop i value becomes
number of characters*/
    return(i);
}
int main()
{
```

```
    int i, j, k;
    char x[20], y[20] = “Ram”;
    printf ("Enter a string”);
    scanf(“%s”, x);
    i = Length(x);
    /* We are sending string variable x for which data is
read */
    j = Length(y);
    /* We are sending string variable for which data is
assigned*/
    k = Length(“Ram”);
    /* We are directly sending the string constant*/
    /* All the three are acceptable forms for this function
call*/
    printf(“Lengths of strings are=%d %d %d\n”, i, j, k);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

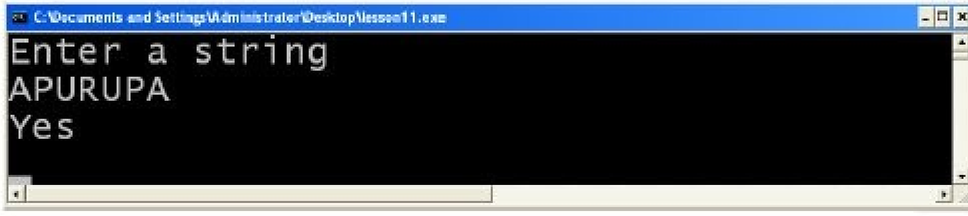


Example 2: ఏదైనా function ఒక string తీసికొని అది palindrome అయితే 1 లేకపోతే 0 రిటర్న్ చేయాలి. ఇందులో పైన రాసిన Length అనే function ను వాడుతున్నాం. ఒక main program దానిని call చెయ్యడం ఎలాగో చూద్దాం.

```
#include<stdio.h>
int length(char x[]) /* Here, x is a string variable*/
{
    int = 0;
    while (x[i]!='\0') i++;
    /* We know after this while loop i value becomes
number of characters*/
    return (i);
}
int Palindrome(char x[] )
```

```
{
    int i, j;
    for( i = 0, j = Length(x) – 1; i < j; i++, j--)
    {
    if(x[i] != x[j]) return(0);
    }
    return( 1 );
}
int main()
{
    char p[20];
    printf("Enter a string\n");
    scanf("%s", p);
    (Palindrome(p))?printf("Yes\n"):printf("No\n");
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



సాధారణంగా variables (scalar variables) function లోనికి వెళ్తే వాటి formal arguments మీద ఏం జరిగినా ఇవి మారవు. అదే arrays అయితే మార్పులు పంపిన arrays లో కనిపిస్తాయి.

Example 3: ఈ function ఒక string, రెండు character టైప్ arguments rc, cr తీసుకొని ఇచ్చిన string లో ఎక్కడ rc ఉంటుందో అక్కడ cr ను పెడుతుంది. ఒక main program దానిని call చెయ్యడం ఎలాగో చూపించడానికి రాశాం.

```
#include<stdio.h>
void REPLACE(char x[], char rc, char cr)
{
    int i = 0;
    while(x[i] != '\0'){
    if(x[i]==rc) x[i] = cr;
    i++;
```

```
    }
}
int main()
{
    char p[20];
    printf("Enter a string\n");
    scanf("%s", p);
    REPLACE(p, 'a', '*');
    printf("%s\n", p);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



Example 4: ఒక function ఒక one dimensional integer array ను, ఒక integer ను తీసుకొని ఇచ్చిన array elements టోటల్ ను రిటర్న్ చేస్తుంది. ఒక main program దానిని call చెయ్యడం ఎలాగో తెలుసుకుందాం.

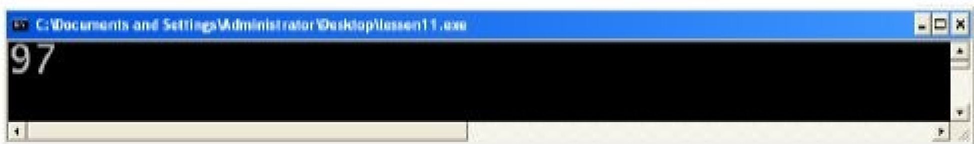
```
#include<stdio.h>
int Sum(int a[], int n )
{
    int i, S = 0;
    for(i =0; i < n; i++) S += a[i];
    return(S);
    }
int main()
{
    int x[20] = { 10, 70, 60, 40, 30, 45 }, S1, S2;
    S1 = Sum( x, 4);
    /* It calculates the sum of first 4 elements*/
    S2 = Sum( x, 6);
    /* It calculates the sum of first 6 elements*/
    printf("Four Elem. Sum=%d\nSix Elem. Sum=%d\n",
S1, S2);
```

```
    return (0);
}
```
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



Example 5: ఒక function ఒక one dimensional integer array ను, ఒక integer ను తీసుకొని ఇచ్చిన array elements లో పెద్ద దాని index ను రిటర్న్ చేస్తుంది.. మరియూ, ఒక main program దానిని call చెయ్యడం ఎలాగో చూద్దాం.

```
#include<stdio.h>
int MI( int a[], int n )
{
    int i, id = 0;
    for(i =1; i < n; i++) if( a[i] > a[id]) id = i;
    return(id);
}
int main()
{
    int x[20] = {10, 17, 20, 56, 97, 32 }, K;
    K = MI (x, 6); /* Function call*/
    printf("%d\n", x[K]);
    return (0);
}
```
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



Example 6: ఒక function ఒక one dimensional integer array, ఒక integer ను తీసుకొని ఇచ్చిన array elements average, maximum, minimum లను రిటర్న్ చేస్తుంది. ఈ మూడింటినీ ఒక dummy array లో స్టోర్ చేస్తాం. మామూలుగా return statement ద్వారా ఒక్క విలువను మాత్రమే రిటర్న్ చేయగలం. అందుకు ఒక dummy array ను

వాడుతున్నాం.ఒక main program దానిని call చెయ్యడం ఎలాగో చూద్దాం.

```
#include<stdio.h>
void STAT1( int a[ ], int n, int b[ ] )
{
    int i, S = a[0], max = a[0], min = a[0];
    for(i =1; i < n; i++){
    S += a[i];
    if(a[i] > max) max = a[i];
    if(a[i] < max) min = a[i];
}
    b[0] = S/n;
    b[1] = max;
    b[2] = min;
}
int main()
{
    int x[ ] = { 70, 71, 56, 97, 66 }, res[3];
    STAT1(x, 5, res);
    /* res array contains average, maximum and
minimum*/
    printf(“Average=%d Maximum=%d Minimum=%d\n”,
res[0], res[1], res[2]);
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



14.1.4. Passing 2-D arrays to functions

return_type function_name(Type arg[ ][5],….)

ఒక function లోనికి ఒక 2-D array argument అని చెప్పడానికి, ఇలా రాస్తాం.

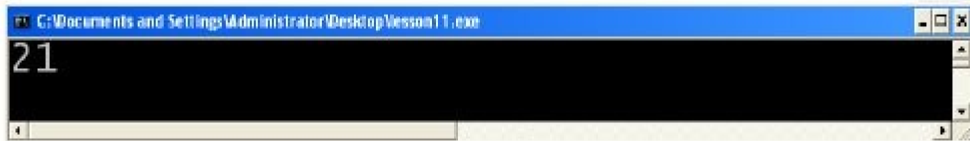
అంటే 2-D array column size తప్పక ఇవ్వాలి. దీని అర్థం, ఆ function ను

call చేస్తున్నప్పుడు ఏ 2-D array అయినా పంపవచ్చు, కాని దాని column size మనం ఇచ్చిన దానితో సమానంగా ఉండాలి.

Example 1: ఇక్కడ ఒక 2-D array ను argument తీసుకొనే function ను రాశాం. ఇది array elements టోటల్ ను రిటర్న్ చేస్తుంది. ఒక main program దానిని call చెయ్యడం ఎలాగో ఈ ప్రోగ్రాం ద్వారా తెలుసుకుందాం.

```
#include<stdio.h>
    int SUM( int x[ ][5], int m, int n)
    {
    int i, j, s = 0;
    for(i = 0; i < m; i++)
    for(j = 0; j < n; j++) s +=x[i][j];
    return(s);
    }
int main()
{
    int a[10][5] = { {1, 2, 3}, {3, 2, 1}, {4, 2, 3} };
    printf("%d\n", SUM(a, 3, 3));
    /* here array a is acceptable as column size matching*/
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.

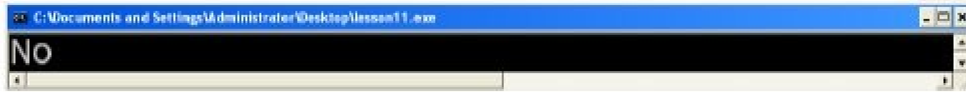


Example 2: ఒక square matrix ను argument గా తీసుకుని అది "Symmetric Matrix" అయితే 1 లేకపోతే 0 రిటర్న్ చేసే function రాస్తున్నాం. ఒక main program దానిని call చెయ్యడం ఎలాగో ఈ ప్రోగ్రాం ద్వారా తెలుసుకుందాం.

```
#include<stdio.h>
    int SYM( int x[ ][5], int n)
    {
    int i, j;
    for(i = 1; i < n; i++)
    for(j = 0; j < i; j++) if(x[i][j]!=x[j][i]) return(0);
    return(1);
    }
```

```
int main()
    {
    int a[10][5] = { {1, 2, 3}, {3, 2, 1}, {4, 2, 3} };
    (SYM(a, 3))?printf("Yes\n"):printf("No\n");
    return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఇలా ఉంటుంది.



14.1.5. Recursive Functions

ఇప్పటి వరకూ ఒక function ను వేరే function లో నుంచి call చేసే functions రాశాం. అలా కాకుండా, ఒక function దానికదే call చేసుకుంటే దానిని recursive function అని అంటాం. దాని గురించి ఇప్పుడు నేర్చుకుందాం.

Example 1: ఈ recursive function రెండు పూర్ణాంకాలు (integers) ను తీసుకుని వాటి product ను రిటర్న్ చేస్తుంది. ఒక main program దానిని call చెయ్యడం ఎలాగో చూద్దాం.

ఇక్కడ, x*y కావాలంటే, y ను x సార్లు add చేయడం అని అనుకున్నాం.

```
#include<stdio.h>
    int prod(int x, int y)
    {
    if(x==1) return y;
    else
    return( y+prod(x-1,y));
    }
int main()
{
    int M, N, result;
    printf("Enter two numbers\n");
    scanf("%d%d", &M,&N);
    result=prod(M,N);
    printf("Result=%d", result);
    return 0
}
```

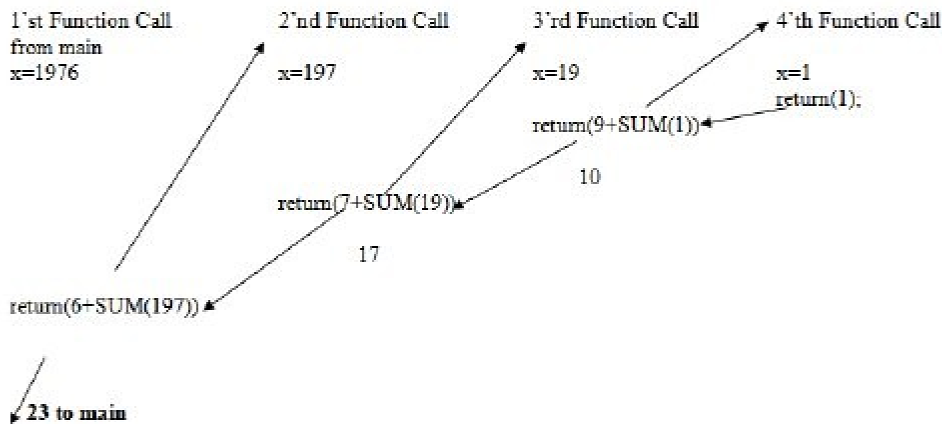
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



Example 2: ఈ recursive function ఒక integer ను argument లాగా తీసుకుని దాని digits మొత్తాన్ని రిటర్న్ చేస్తుంది. ఒక main program దానిని call చెయ్యడం ఎలాగో చూద్దాం.

```
#include<stdio.h>
int SUM(int x)
    {
    if(x/10) return(x%10+ SUM(x/10));
    else return(x);
    }
int main()
    {
    int Num;
    printf("Enter an integer\n");
    scanf("%d", &Num);
    printf("Digits sum=%d\n", SUM(Num));
    return 0;
}
```

ఈ కింది Num విలువ 1976 అయినప్పుడు function call ఎలా పని చేస్తుందో చూపించాం.



పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



Example 3: ఈ recursive function ఒక integer ను తీసుకుని దాని factorial value ను రిటర్న్ చేస్తుంది. ఒక main program దానిని call చెయ్యడం ఎలాగో చూద్దాం.

$$\lfloor 5 = 1.2.3.4.5 = \lfloor 4.5$$
$$\lfloor 4 = 1.2.3.4 = \lfloor 3.4$$
$$\lfloor 3 = 1.2.3 = \lfloor 2.3$$

పై analysis నుంచి మనం ఈ కింది విధమైన statement ఇవ్వవచ్చు,

$$\lfloor n = \lfloor (n-1).n$$
$$\lfloor 0 = 1$$

అంటే, n factorial కావాలంటే (n-1) factorial కనుక్కొని దానిని n తో గుణిస్తే సరిపోతుంది. అలాగే, n-1 factorial కావాలంటే (n-2) factorial కనుక్కొని దానిని (n-1) తో గుణిస్తే సరిపోతుంది.

```
#include<stdio.h>
long fact(int n)
    {
    if(n==0) return 1;
    else
    return (n*fact(n-1));
    }
int main()
    {
    int Num;
    printf("Enter an integer\n");
    scanf("%d", &Num);
    printf("Factorial Value of "%ld=%ld\n", fact(Num));
    return 0;
}
```

ఈ కింది టేబుల్ n=4 అయినప్పుడు, recursive function ఎలా పని చేస్తుందో చూపిస్తుంది.

1'st Function Call	2'nd Function Call	3'rd Function Call	4'th Function Call	5'th Function Call
n=4	n=3	n=2	n=1	n=0
				return 1;
			return(1*fact(0)); 1	
		return(2*fact(1)); 1		
	return(3*fact(2)); 2			
return(4*fact(3)); 6				

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter an integer
4
Factorial value of 4=24
```

14.1.6. Storage Classes

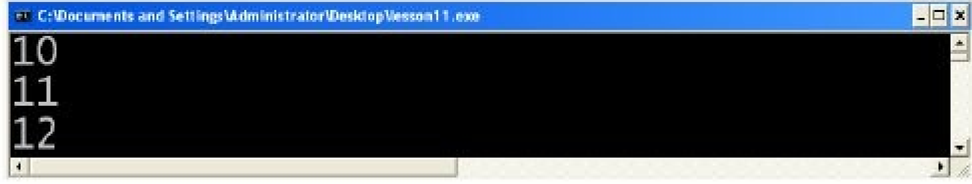
Variables ను వాటి లైఫ్, స్కోప్ ను బట్టి ఈ కింది విధంగా విభజిస్తారు. లైఫ్ అంటే వాటికి allocate చేసిన memory ఎంత వరకు ఉంటుంది. స్కోప్ అంటే వాటిని ఎక్కడ వాడుకోవచ్చు.

Example 1: ఈ కింది ప్రోగ్రాంలో static తీసివేసి ఫలితాన్ని గమనించండి.

```
#include<stdio.h>
void F()
    {
    static int a=10;
    printf("%d\n",a++);
    }
int main()
{
```

```
    F();
    F();
    F();
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



14.1.7. Macros

ఒక function ను call చేసినప్పుడు, కొంత సమయం వేచి ఉండాలి . ఒక్కోసారి, function లోపల చేసే కాలిక్యులేషన్లకు పట్టే సమయం ఎక్కువైనట్లయితే function బదులు macro రాస్తాం (ఉదాహరణకు, ఒక రూపాయి money order ద్వారా పంపాలంటే 18 రూపాయలు ఖర్చవుతుంది. అప్పుడు, మనం ఏమి చేస్తాం? ఒక రూపాయి కవరు లో పెట్టి పోస్ట్ చేస్తాం). ఇంతకు ముందు మనం వాడిన, isupper, islower, toupper అనేవి macros.

Macros వాడిన చోట వాటి బాడీ (definition) substitute అవుతుంది. ఒక్కోసారి, ఇది మన టైపింగ్ భారాన్ని తగ్గిస్తుంది. అలాగే, macros లో టైప్ చెకింగ్ వీలు కాదు.

Example 1: ఈ ప్రోగ్రాంలో LOOP అనే macro ను define చేసి దానిని main లో వాడాం. ఎక్కడ LOOP వాడామో అక్కడ for(i=0;i<n;i++) substitute అయి ఆ తర్వాత ప్రోగ్రాం కంపైల్ అవుతుంది.

```
#include<stdio.h>
#define LOOP for(i=0;i<n;i++)
int main()
{
    int i,n, a[10];
    printf("Enter number of integers \n");
    scanf("%d",&n);
    printf(Enter %d integers in to the array \n,n");
    LOOP
    scanf("%d",&a[i]);
    printf("numbers given are\n");
    LOOP
    printf("%d\n", a[i]);
    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter number of  integers
5
Enter 5 integers in to the array
33 12 22 31 33
Numbers given are
33
12
22
31
33
```

Example 2: ఈ ప్రోగ్రాంలో కూడా LOOP అనే macro ను define చేసి దానిని main లో వాడాం. కానీ, ఇందులో దానికి కొన్నింటిని arguments గా ఇచ్చాం. LOOP ఎక్కడ వాడామో అక్కడ for(i=0;i<n;i++) substitute అయిన తర్వాత ప్రోగ్రాం కంపైల్ అవుతుంది. కానీ ఏ argument స్థానంలో ఏది పంపితే అది substitute అవుతుంది.

```
#include<stdio.h>
#define LOOP(i,n) for(i=0;i<n;i++)
int main()
{
    int i,k,n, a[10];
    printf("Enter an integer\n");
    scanf("%d", &n);
    LOOP(i,n) /* for(i=0; i<n; i++) is substituted */
    scanf("%d", &a[i]);
    LOOP(k,n) /* for(k=0; k<n; k++) is substituted */
    printf("%d\n", a[k]);
    return(0);
}
```

Example 3: ఈ ప్రోగ్రాం పైన ఇచ్చిన macro ను వేరొక ప్రోగ్రాంలో ఎలా వాడాలో చూపిస్తు0 ది.

```
#include<stdio.h>
#define LOOP(i,n) for(i=0;i<n;i++)
```

```
int main()
{
    int i,j,n, a[10][10];
    printf("Enter size of square matrix\n");
    scanf("%d", &n);
    printf("Enter the data\n");
    LOOP(i,n) /* for(i=0; i<n; i++) is substituted */

    LOOP(j,n) /* for(j=0; j<n; j++) is substituted */
    scanf("%d", &a[i][j]);
    "printf("Printing the data\n");"
    LOOP(i,n)
    { /* for(i=0; i<n; i++) is substituted */
    LOOP(j,n) /* for(j=0; j<n; j++) is substituted */
    printf("%d\t", a[i][j]);
    printf("\n");
    }
return(0);
}
```

Example 4: ఈ ప్రోగ్రాంలో ఒక macro definition ఒక్క line కన్నా ఎక్కువ అయితే ఎలాగో చూపిస్తుంది. చివరి లైనుకు తప్ప మిగతా అన్నింటికి చివరలో back slash (\) ఇవ్వాలి.

```
#include<stdio.h> "printf("Printing the data\n");"
#define LOOP(i,j,n) for(i=0;i<n;i++)\
    for(j=0;j<n;j++)
    int main(){
    int i,j,n, a[10][10];
    printf("Enter size of square matrix\n");
    scanf("%d", &n);
    printf("Enter the data\n");
    LOOP(i,j,n) /* Two lines of the macro are substituted*/
    scanf("%d", &a[i][j]);
    LOOP(i,j,n) /* Two lines of the macro are substituted*/
    printf("%d\t", a[i][j]);
    return(0);
}
```

Example 5: ఈ ప్రోగ్రాం 5, 4 లను ఫలితంగా ఇస్తుంది. అనలైజ్ చేయండి. మీకే తెలుస్తుంది.

```
#include<stdio.h>
#include<stdio.h>
#define norm(a,b) sqrt(a*a+b*b)
int main()
{
    float p=3,q=4,r,s;
    r=norm(p,q); /* r=sqrt(p*p+q*q);*/
    s=norm(p+1,q+1); /* s=sqrt(p+1*p+1+q+1*q+1);*/
    printf("%f %f\n",r,s);
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



Conclusions

ఈ పాఠంలో functions రాయడమెలాగో నేర్చుకొన్నాం. అలాగే recursive functions, macros, storage classes, గరించి కూడా తెలుసుకున్నాం.

Pointers

15. Pointers

15.1. Introduction

సాధారణమైన variable లానే, Pointer కూడా ఒక variable, కానీ దీనికి address మాత్రమే (assign) ఇవ్వగలం. ఒక variable declaration statement లో variable కు ముందు asterisk(*) ఉంటే దాని pointer type variable అని చెప్పినట్లు అర్ధం వస్తుంది. ఉదాహరణకు.

```
    int *number;
    char *ch;
    float *gnumber;
```

పైన చూపిన విధంగా, pointer variables లో types ఉంటాయి. ఏ టైప్ అయినా, వాటికి మనం address మాత్రమే ఇచ్చేవీలుంటుంది. Pointer definition ఇంకా లోతుగా చెప్పాలంటే, దానికి మనం allocate అయిన

memory address మాత్రమే ఇవ్వగలం. మనకు నచ్చిన నంబరును ఇవ్వలేం,'0' ను తప్ప. అంతే కాకుండా, ఒక pointer variable కు అదే టైప్ variable address మాత్రమే ఇవ్వాలి. ఉదాహరణకు,

```
int k=10, *ptr;
ptr=&k;
```

ఇక్కడ, variable k address ను pointer variable ptr కు ఇచ్చాం. ఇది valid ఆపరేషన్. అప్పుడు, ptr ను k కు pointer అని అంటారు. k విలువను ptr ద్వారా కూడా access చెయ్యవచ్చు. అంటే, ptr కు "indirection operator" (asterisk) ను జోడిస్తే, *ptr అనేది k అవుతుంది. ఎక్కడైతే k ను వాడాలో అక్కడ *ptr ను వాడవచ్చు. ఉదాహరణకు, k కు 7 ఇవ్వాలంటే, *ptr = 7; అని రాయవచ్చు. అప్పుడు, pointer ptr విలువ ఏ memory ని point చేస్తుందో (అది ఇప్పుడు k యొక్క memory), దానిలో 7 స్టోర్ చేస్తుంది. ఈ కింది printf statement, k విలువని *ptr ద్వారా ప్రింట్ చేస్తుంది.

```
printf("%d\n",*ptr);
```

ముఖ్య గమనిక - Declaration statements లో ఒక variable కు ముందు * ఉంటే అది pointer type అని, ఇంకేదైనా statement లో ఒక pointer variable కు ముందు * ఉంటే, indirection operator అని అంటారు. అది ఆ pointer variable point చేస్తున్న memory లో ఉన్న విలువను indicate చేస్తుంది.

Example 1: ఈ కింది ప్రోగ్రాం ద్వారా పైన చెప్పిన concepts ను ఎలా ఉపయోగించాలో చూద్దాం.

```
#include<stdio.h>
int main()
{
    int *p, i=20;
    p=&i;
    i++;
    printf("%d %d\n", i, *p);
    i=i+10;
    printf("%d %d\n", i, *p);
    *p=*p+10;
    printf("%d %d\n", i, *p);
    (*p)++;
    printf("%d %d\n", i, *p);
    printf("Enter a value for i\n");
    scanf("%d", p);
    printf("%d %d\n", i, *p);
```

```
    return 0;
}
```

Memory of main

Address	Memory	Variable
0023FE66		p
0023FE70	20	i

int *p, i=10; అనే statementను execute చేసిన తర్వాత.

Memory of main

Address	Memory	Variable
0023FE66	0023FE66	p
0023FE70	20	i

p=&i; అనే statementను execute చేసిన తర్వాత.

Memory of main

Address	Memory	Variable
0023FE66	0023FE66	p
0023FE70	21	i

i++; అనే statementను execute చేసిన తర్వాత.

Memory of main

Address	Memory	Variable
0023FE66	0023FE66	p
0023FE70	31	i

i=i+10; అనే statementను execute చేసిన తర్వాత.

Memory of main

Address	Memory	Variable
0023FE66	0023FE66	p
0023FE70	41	i

*p=*p+10; అనే statementను execute చేసిన తర్వాత.

Memory of main

Address	Memory	Variable
0023FE66	0023FE66	p
0023FE70	42	i

(*p)++; అనే statementను execute చేసిన తర్వాత.

Memory of main

Address	Memory	Variable
0023FE66	0023FE66	p
0023FE70	566	i

scanf("%d", p); అనే statementను execute చేసిన తర్వాత మరియూ user 566 input లాగా ఇచ్చాడు అని అనుకోంటూ.

పైన ఇచ్చిన టేబుల్ , పై ప్రోగ్రాంఎలా పని చేస్తుందో ఉదాహరణకు కొన్ని డమ్మీ విలువలను తీసుకుని చూపిస్తుంది. ఇక్కడ memory address లు అన్నీ డమ్మీ విలువలు.

Output:

```
21 21
31 31
41 41
42 42
Enter a value for i
566
566 566
```

Example 2: ఈ ప్రోగ్రాంలో pointer variable విలువను ఇంకొక pointer variable కు ఇవ్వవచ్చు అని చెప్పడానికి.

```
#include<stdio.h>
void main()
{
```

```
    int I=10,J=17, *p, *q;
    p=&I;
    q=p;
    printf("%d %d\n", *p, *q);
    p=&J;
    printf("%d %d\n", *p , *q);
}
```

Memory of main

Address	Memory	Variable
0023FF66	10	I
0023FF68	17	J
0023FF70		p
0023FF74		q

Soon after declaration

Memory of main

Address	Memory	Variable
0023FF66	10	I
0023FF68	17	J
0023FF70	0023FF66	p
0023FF74		q

After executing p=&I

Memory of main

Address	Memory	Variable
0023FF66	10	I
0023FF68	17	J
0023FF70	0023FF66	p
0023FF74	0023FF66	q

After executing q=p;

After executing
printf("%d %d\n", *p, *q);
స్క్రీన్ మీద 10 మరియూ 10 వస్తాయి.

Memory of main

Address	Memory	Variable
0023FF66	10	I
0023FF68	17	J
0023FF70	0023FF68	p
0023FF74	0023FF66	q

After executing p=&J;

After executing
printf("%d %d\n", *p, *q);
స్క్రీన్ మీద 17 మరియూ 10 వస్తాయి.

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

C:\Documents and Settings\Administrator\Desktop\lesson11.exe

```
10    10
17    10
```

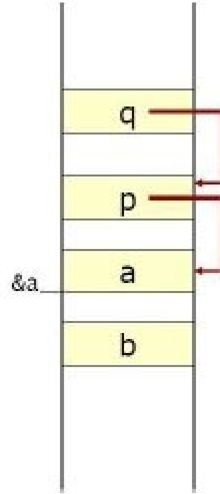
15.2. Pointer to Pointer

ఒక pointer variable కు కూడా pointer variable ను declare చెయ్యవచ్చు. ఈ విధంగా చేయడానికి మనకు pointer to a pointer టైప్ variable, అంటే, int **a టైప్ వాడాల్సి ఉంటుంది. ఈ టైప్ variable లో ఒక

integer టైప్ pointer variable యొక్క address ను స్టోర్ చేయవచ్చు.

```
int   a   // variable
int   b
int *p    // pointer to int
int **q   // ptr to ptr to int

a =  7    // a=7
p = &a    // p points to a
q = &p    // q points to p
b = *p    // a=7, b=7
*p = 5    // a=5, b=7
**q = 3   // a=3, b=7
```



పై code fragment లో రెండు integer variables a, b లను, ఓ integer pointer, p, ఒక integer point to pointer, q ను declare చేసాం. తర్వాత, a కు 7 ఇచ్చాం, a address ను p కు ఇచ్చాం. ఆ తర్వాత, q కు p address ను ఇచ్చాం. తర్వాత, b=*p execute అయినప్పుడు, a విలువ, 7, b కు ఇవ్వబడినది. అలాగే, *p=5 statement వలన, a విలువ 5 అయింది. (b విలువ మారలేదు, అది ఇంకా 7). ఆ తర్వాత, **q=3 statement execute అవడంతో a విలువ 3 అవుతుంది. ఎలా అంటే, p లో a address ఉంది, అలాగే q లో p యొక్క address ఉంది. అందువలన **q లో *q అంటే p అవుతుంది. ఇప్పుడు మరొక indirection అప్లై చేస్తే, *p, అంటే a అని అర్థం.
ఈ pointer to pointer concept ను ఎంతవరకైనా పొడిగించవచ్చు.
ఉదాహరణకు,

```
    int **a;
    int ***a;
    int ****a;
```

Example 3: ఈ ప్రోగ్రాం ఫలితం ఏమిటో చూడండి.

```
#include<stdio.h>
int main()
{
    int *p, i=10;
    p=&i;
    printf("\n%d %d\n",*p,
*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&i);
    return 0;
}
```

Output: 10 10.

ఒక variable i address &i, అలాగే, *&i అంటే i విలువ. పై ప్రోగ్రాంలో దీనిని recursively వాడాం. పైన ఇచ్చిన entire expression i విలువను తెలుపుతుంది.

15.3. Passing Pointers to Functions

ఒక function లోపలికి address పంపుతున్నామని చెప్పడానికి formal argument ముందు * ఈ కింది విధంగా పెడతాం.

```
    void xyz(int *x);
```

అంటే, పై function ను call చేస్తున్నపుడు ఒక integer variable address పంపాలి, ఈ function లో x ఒక pointer type variable. ఈ function లో x pointer variable ను వాడి ,అది point చేస్తున్న address లో ఏం చేసినా అది మనం పంపిన actual argument మీద జరుగుతుంది. దానినే Passing by address అని అంటారు.

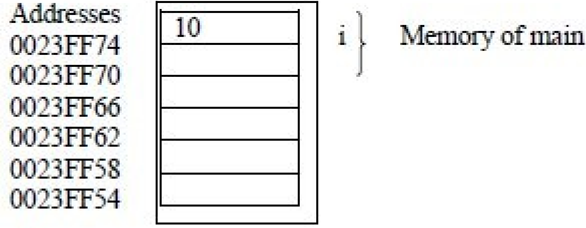
గమనిక: ఒక function ను call చేసినప్పుడు అందులో ఉన్న formal arguments కు, లోపల declare చేసే variable (scratch variables)కు మెమొరీ allocate అవుతుంది. Function లోనుంచి రిటర్న్ అవగానే ఈ మెమొరీ de-allocate అవుతుంది. ఈ allocate చేసిన మెమొరీని actiactivation frame లేదా activation record లేదా stack frame లేదా stack record □□□అంటారు.

Example 4: ఈ ప్రోగ్రాంలో, main లో ఉన్న i అనే variable address ను xyz function కు పంపిస్తున్నాం. ఆ address లో xyz function లో 100 స్టోర్ చేస్తున్నాం. Function లోపలి నుంచి రాగానే x విలువ ను main లో ప్రింట్ చేస్తే 100 గా ప్రింట్ అవుతుంది. అలా Function లో చేసిన మార్పు main లో 'i' variable లో మనం చూడగలుగుతున్నాం.

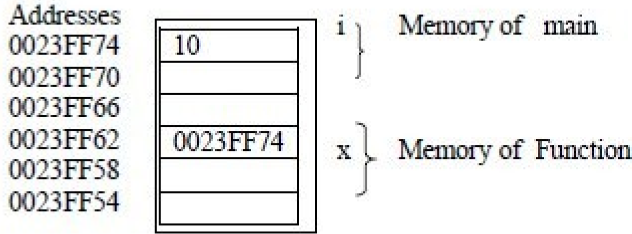
```
#include<stdio.h>
void xyz(int *x)
{
    *x=100; /* We are storing 100 in the memory refered 
by x*/
}
int main()
{
    int i=10;
    printf("Value of i before function call=%d\n", i);
    xyz(&i);/* The function prototype indicates that we 
have to send address. We are sending the address of i*/
    printf("Value of i before function call=%d\n", i);
```

```
    return 0;
}
```

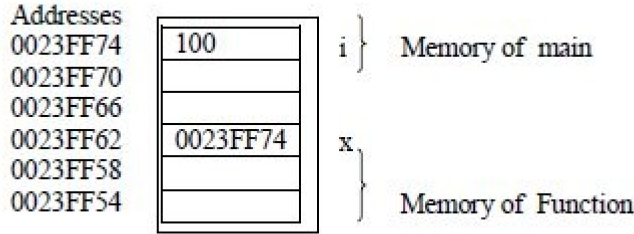
Function Call చేయకముందు memory ఇలా ఉంటుంది.



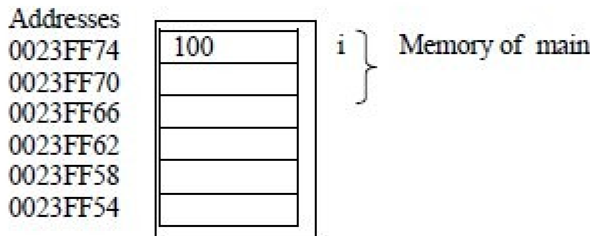
ఇప్పుడు, xyz(&i); అనేది execute అయితే memory ఇలా ఉంటుంది.



ఇప్పుడు *x=100; అనేది execute అయితే 100 ను 0023FF74 లో స్టోర్ చేస్తున్నాం, అంటే i లో.



Function లో నుంచి control main లోపలికి రాగానే memory ఈ కింది విధంగా ఉంటుంది:



పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Value of i before function call=10
Value of i after function call=100
```

Example 5: ఈ ప్రోగ్రాంలో ఒక integer variable address తీసుకొని అందులో ఉండే విలువను reverse చేసే function ను రాసి దానిని main నుంచి

call చేయడం ఎలాగో చూద్దాం.

```
#include<stdio.h>
void REV(int *x)
{
    int n, s=0;
    n=*x;
    while(n)
    {
    s=s*10 + n%10;
    n=n/10;
    }
    *x=s;
}
int main()
{
    int i;
    printf("Enter a Number\n");
    scanf("%d", &i);
    REV(&i);
    printf("Value of the number after function call=%d\n",
i);
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

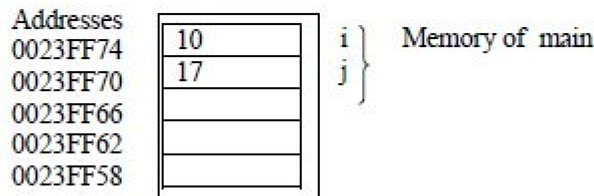


Example 6: రెండు సంఖ్యలున్న అంకెను తీసుకుని దాని అంకెల స్థానాలను తారుమారు చేసి call చేసే విధంగా ఒక ప్రోగ్రాం రాయండి.

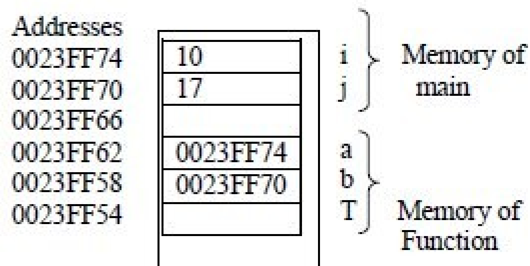
```
#include<stdio.h>
void swap(int *a, int *b)
{
    int T;
    T=*a;
    *a=*b;
    *b=T;
```

```
}
int main()
{
    int i, j;
    printf("Enter two Numbers\n");
    scanf("%d%d", &i, &j);
    swap(&i,&j);
    printf("Values after function call=%d %d\n", i, j);
    return 0;
}
```

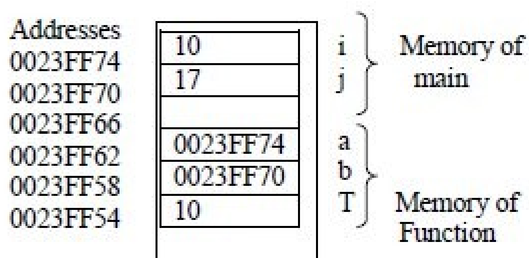
main program లో variables declare చేసిన తర్వాత memory content.



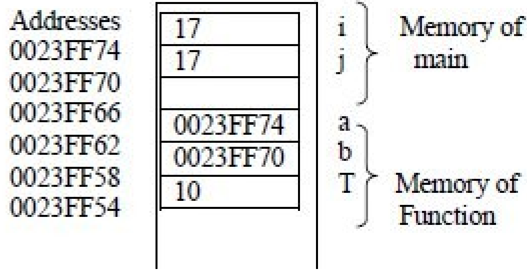
swap() function call చేసిన తర్వాత memory content. అంటే, i, j variables address లు a,b అనే variables కు assign అవుతాయి.



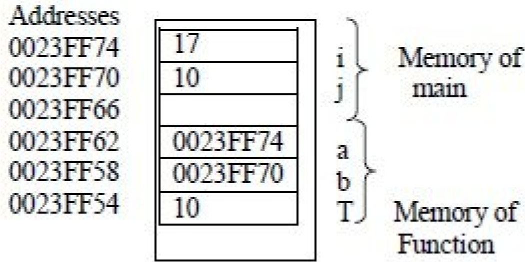
T=*a; అనే statement execute చేసిన తర్వాత memory content. ఇక్కడ, pointer variable a పాయింట్ చేస్తున్న మెమొరీలో ఉన్న విలువ, అంటే 10, T కు assign అవుతుంది.



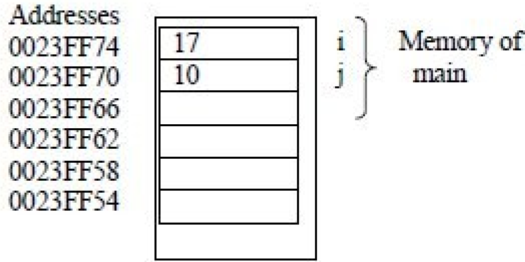
*a=*b; అనే statement execute చేసిన తర్వాత memory content. ఇక్కడ, pointer variable b పాయింట్ చేస్తున్న మెమొరీలో ఉన్న విలువ, అంటే 17, pointer variable a పాయింట్ చేస్తున్న మెమొరీలో ఉన్న విలువ IA లా అవుతుంది



*b=T; అనే statement execute చేసిన తర్వాత memory content.
ఇక్కడ, pointer variable b పాయింట్ చేస్తున్న మెమొరీలో ఉన్న విలువ IA లాగా T, అంటే 10 అవుతుంది



swap() function IO లో నుంచి main లోకి వచ్చిన తర్వాత memory content.



ఇలాగా i, j అనే variables విలువలు exchange అవుతాయి.
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter two Numbers
10 17
Values after function call=17   10
```

Example 7: ఇక్కడ రాసిన function ఒక 1-D integer array ను తీసుకొని దాని elements average, maximum, minimum లను రిటర్న్ చేస్తుంది.
Solution: మనకు మూడు విలువలు ఈ function నుంచి రిటర్న్ అవ్వాలి. కనుక మనము array తో పాటూ మూడు variables address లు కూడా main నుంచి పంపుతాం. Function లో ఆ address లో మనం రిటర్న్ చేయాలనుకున్న average, maximum, minimum లను స్టోర్ చేస్తాం. అలా మనకు కావలసిన మూడు విలువలు main లో నుంచి వేటి address లు పంపామో వాటిలోకి వస్తాయి.
void stat3(int a[], int n, int *X, int *Y, int *Z)

```
{
    int I,s,max,min;
    for(I=1,s=a[0],max=a[0],min=a[0];I<n;I++){
    s+=a[I];
    if(max<a[I])max=a[I];
    if(min>a[I])min=a[I];
}
    /*storing the values to be returned in the addresses*/
    *X=s/n; *X=max; *Z=min;
}
int main()
{
    int N, a[10],I,A,B,C;
    printf("Enter number of elements \n");
    scanf("%d", &N);
    /* reading elements into array */
    printf("Enter elements values \n");
    for(I=0;I<N;I++) scanf("%d", &a[I]);
    stat3(a, N, &A, &B, &C);
    printf("Average=%d\nMaximum=%d\nMinimum=
%d\n", A,B,C);
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter  Number  of elements
5
Enter elements values
90
90
78
17
56
Average=90
Maximum=2009291924
Minimum=17
```

15.4. Dynamic Memory Allocation

సీ లాంగ్వేజీ లో ఉన్న arrays (int a[10]) ఉపయోగించి రాసిన ప్రోగ్రాంల్లో flexibility ఉండదు. flexibility కావాలంటే memory నిరుపయోగం అవుతుంది. ఈ రెండు సమస్యలను అధిగమించడానికి మనం dynamic arrays వాడతాం. అంటే, ప్రోగ్రాం రన్ అవుతున్నపుడు ఎన్ని elements ఉండే array కావాలో అంతదే create చేయవచ్చు.ఈ విధంగా create చేసిన arrays memory ఎప్పుడైనా de-allocate చెయ్యవచ్చు. అదే లాంగ్వేజి లో ఉన్న arrays memory ని మనం de-allocate చెయ్యలేం.

memory management కోసం వాడే Functions వివరాలు.

1. void * malloc(int n)

ఈ function ఒక integer (n)ను argument గా తీసుకొని అన్ని bytes ఉండే memory ని allocate చేసి దాని starting address (generic address) ని రిటర్న్ చేస్తుంది. memory allocate చేయలేకపోతే ఇది null రిటర్న్ చేస్తుంది.

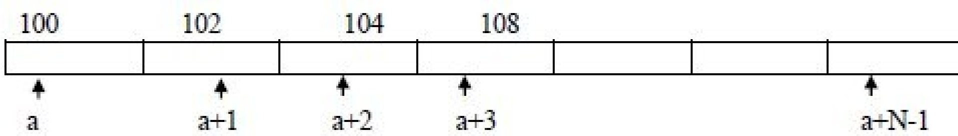
2. free(address)

ఈ function లోకి ఒక address పంపిస్తే ఆ address నుంచి మనకు allocate చేసి ఉన్న memory de-allocate అవుతుంది.

1-D Dynamic Array Creation

```
int *a, N, i,j;
printf("Enter required number of elements of array to be created \n");
scanf("%d", &N);
/* allocating memory for the array */
a=(int *) malloc(N* sizeof(int));
```

పైన N* sizeof(int) అనేది మనకు కావాల్సిన array కు కావల్సిన memory(bytes లో). Function malloc అన్ని bytes ఉండే memory allocate చేసి దాని address ని void *(generic address)గా ఇస్తుంది. దానిని integer address లోనికి int * వాడి typecast చేసి integer pointer a కు ఇస్తున్నాం. ఈ memory ని మనం ఎలా కావలంటే అలా చూడవచ్చు. ఒక float array గా చూడాలంటే వచ్చిన address ను float * లోనికి typecast చెయ్యాలి. ఒక double array లాగా చూడాలంటే వచ్చిన address ను double * లోకి typecast చెయ్యాలి. ఇక్కడ, a అనే దానిని first element కి సంబంధించిన address అని అనుకోవచ్చు. అలాగే, a+1, a+2, …అనేవి పక్క elements address లు అవుతాయి. మన కంప్యూటరులో integer కు 2 bytes అని, మొదటి element address 100 అని అనుకుంటే, పక్కపక్క elements address లను ఇలా చూపించవచ్చు.



గమనిక: మన కంప్యూటరులో float variable కు 4 bytes allocate అయితే, a ఒక float pointer అయితే, a విలువ x అయితే a+6 విలువ x+6*4 అవుతుంది.
double variable కు 8 bytes allocate అయితే, a ఒక double pointer అయితే, a విలువ x అయితే a+6 విలువ x+6*8 అవుతుంది.
char variable కు 1 byte allocate అయితే, a ఒక character pointer అయితే, a విలువ x అయితే a+6 విలువ x+6 అవుతుంది. ఇలా pointers టైప్ పని చేస్తుంది. అలాగే, a కనుక float pointer అయితే, a++ (లేక ++a) అయిన తర్వాత దాని విలువ 4 పెరిగి ఉంటుంది. అలాగే, ఇక్కడ 'a' float pointer అయితే, a-- (లేక --a) అయిన తర్వాత దాని విలువ 4 తగ్గి ఉంటుంది. ఇవి వేరే టైప్ pointers variables కూడా వర్తిస్తాయి.

Reading the data into array

పైన చెప్పిన విధంగా, a+i అనేది మనం create చేసిన array లోని ith element address. కాబట్టి దానిని వాడి scanf ద్వారా array లోపలికి డేటా ఇలా రీడ్ చేస్తాం.

```
for(i=0;i<N;i++) scanf("%d", a+i );
```

Printing the data of the array

అలాగే, a+i అనేది మనం create చేసిన array లోని ith element address అయితే *(a+i) అనేది దాని విలువను indicate చేస్తుంది. ఇక్కడ కూడా a[i] వాడవచ్చు. ఈ కింద ఇచ్చినది, array లో ఉన్న విలువల ను ప్రింట్ చేస్తుంది.

```
    for(i=0;i<N;i++) printf("%d %d\n", *(a+i), a[i] );
```

Example 8: పైన చెప్పినవన్నీ కలిపి ఉన్న ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
#include<stdib.h>
int main()
{
    int *a,n,i;
    printf("Enter the size of the array:");
    scanf("%d",&n);
    a=(int*)malloc(n*sizeof(int)); /* Allocating memory */
    printf("Enter elements into array\n");
    for(i=0;i<n;i++) scanf("%d",a+i); /* Reading data*/
    printf("Array elements are \n");
    for(i=0;i<n;i++)printf(" %d",*(a+i)); /* Displaying the data */
    free(a); /* to de-allocate memory*/
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter the size of the array:5
Enter elements into array
99
12
33
90
33
Array elements are
  99  12  33  90  33_
```

Example 9: ఈ ప్రోగ్రాం pointers మీద increment, decrement, relational operators ఉపయోగించటాన్ని చూపిస్తుంది. టోటల్ మీద, మనం create చేసి రీడ్ చేసిన array విలువ reverse అయిన తర్వాత ప్రింట్ చేస్తుంది.

```
#include<stdio.h>
#include<stdib.h>
void swap(int*a,int*b)
{
    int T,
    T=*a;
    *a=*b;
    *b=T;
}
int main()
{
    int *a,n,i,*p,*q;
```

```
    printf("Enter the size of the array:\n");
    scanf("%d",&n);
    a=(int*)malloc(n*sizeof(int));
    printf("Enter elements into array\n");

    for(i=0;i<n;i++) scanf("%d",a+i);
    p=a;
    q=a+n-1;

    while(p<q) swap(p++,q--);

    /* Initially, q are pointers to first and last elements respectively (say
for example 100, 114 respectively). In next loop they are pointers to
second and last but one elements. Each time their addresses are sent to
SWAP function which exchanges the values at those addresses. Thus
entire array values get reversely stored. */
    printf("Array elements are \n");
    for(i=0;i<n; i++) printf(" %d",*(a+i));
    return 0;
}
```

ఇనీషియలు dynamic array స్టేటస్.

100	102	104	108	110	112	114
1	2	4	8	8	9	**2**

a,p q

మొదటి while loop iteration అయిన తర్వాత.

100	102	104	108	110	112	114
2	**2**	4	8	8	**9**	1

a,p q

రెండవ while loop iteration అయిన తర్వాత.

100	102	104	108	110	112	114
2	9	**4**	8	**8**	2	1

a,p q

మూడవ while loop iteration అయిన తర్వాత.

100	102	104	108	110	112	114
2	9	8	8	4	2	1

a,p q

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter the size of the array:
7
Enter elements into array
1  2 4 8  8  9 2
Array elements are
  2  9  8  8  4  2  1_
```

15.04.1 Dangling Memory

Memory మన కోసం allocate అయి మనం వాడుకోలేని mmemory ని Dangling memory అని అంటారు. ఇది సాధారణంగా జరిగే తప్పు. ఒక function లో dynamic array create చేసి, దాని address ను రిటర్న్ చేయకుండా బయటకు వస్తే, అది dangling memory అవుతుంది. ఆ function లో create చేసిన dynamic array ను ఏ function అయినా వాడుకోవాలంటే దాని address తెలిస్తేనే వీలవుతుంది. అంటే ఆ function కనుక address రిటర్న్ చేస్తే dangling memory లేకుండా చేసినట్లు అవుతుంది. ఈ కింది ప్రోగ్రాంలో, return p; లేకపోతే dangling memory సమస్య ఉన్నట్లు.

Example 10:

```
#include<stdio.h>
#include<string.h>
char* RS()
{
    char x[80], *p;
    printf("Enter a string\n");
    scanf("%s", x);
    p=(char *) malloc(strlen(x)+1);
    strcpy(p,x);
    return p;
}
int main()
{
     char *T;
    T=RS();
     printf(" Given string = %s\n", T); /* Given string will be printed
now*/
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter a string
Rama
Given string is=Rama
```

15.04.2 Dangling Pointer

ఏదైనా pointer, మన account లో లేని memory ని point చేస్తూంటే దానిని dangling pointer అని అంటారు. అది point చేస్తున్న memory ని దీని ద్వారా access చేస్తే మనకు run-time error వస్తుంది. ఇది కూడా మన తప్పు చేయడం వల్ల వస్తుంది. pointer variable ను free() function లోకి పంపి memory ని free చేసినా, pointer variable విలువ అదే ఉంటుంది. దీన్ని సరిదిద్దాలంటే, free function call అయిన వెంటనే, pointer variable కు సున్నా ఇవ్వాలి.

Example 11: ఈ ప్రొగ్రాంలో ఒక function లోకి 1-D integer array పంపి elements average, maximum, minimum లను రిటర్న్ చేస్తుంది. దీనికి గానూ ఒక dynamic array ను function లో create చేసి మనం రిటర్న్ చేయాల్సిన మూడు విలువలను అందులో పెట్టి dynamic array కి సంబంధించిన starting address రిటర్న్ చేస్తున్నాం.

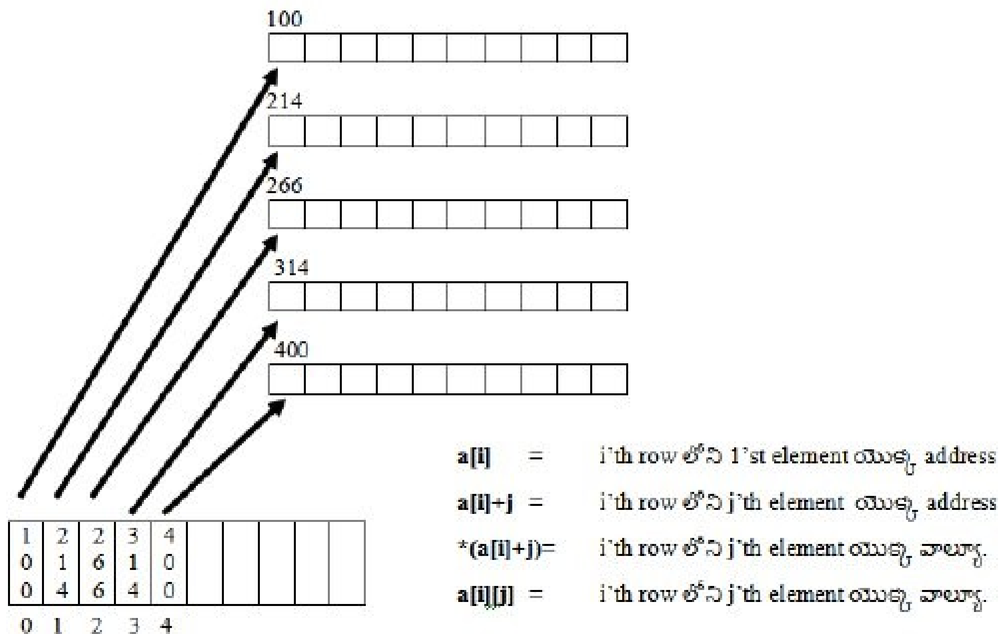
```
int * stat4(int a[], int n)
{
    int I,s,max,min,*p;
    for(I=1,s=a[0],max=a[0],min=a[0];I<n;I++)
    {
    s+=a[I];
    if(max<a[I])max=a[I];
    if(min>a[I])min=a[I];
    }
    p=(int* ) malloc(3*sizeof(int));
    p[0]=s/n; *(p+1)=max; p[2]=min;
    /* Three values are stored in dynamic array p */
    return (p); /* address of the dynamic array is returned*/
}
int main()
{
    int N, a[10],I,*RES;
    printf("Enter a Number\n");
    scanf("%d", &N);
    /* reading elements into array */
    for(I=0;I<N;I++) scanf("%d", &a[I]);
    /* calling the function*/
    RES=stat4(a, N);
    /* RES becomes pointer to the array created in function which is
having maximum, minimum and average */
    printf("Average=%d\n Maximum=%d\n Minimum=%d\n",
RES[0],RES[1],RES[2]);
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter  Number  of elements
5
Enter elements values
90
90
78
17
56
Average=90
Maximum=2009291924
Minimum=17
```

15.04.3 Creating 2D Arrays Flexibly (Array of pointers approach)

మనం1-D arrays ను ఎలా flexible గా create చెయ్యాలో నేర్చుకున్నాం. ఇప్పుడు 2-D arrays ను ఎలా create చెయ్యాలో నేర్చుకుందాం. ముందుగా మనకు కావలసిన టైప్ లో ఉన్న ఒక pointer array (ఉదాహరణ: int *a[10])ను తీసుకుంటాం. ఆ తర్వాత మనకు కావల్సిన 2-D array కి సంబంధించిన rows (1-D arrays) ను malloc ద్వారా create చేస్తూ, వచ్చిన ప్రతీ 1-D array address (starting element addresses)ను pointer array లో స్టోర్ చేస్తాం. ఈ address లు ద్వారా మనం create చేసిన 2-D array లోని ఏ element అయినా access చేయవచ్చు. ఈ కింద ఇచ్చిన బొమ్మను, example program ను చూడండి.



Example 12: ఈ ప్రోగ్రాంఒక 2-D array ను array of pointers ద్వారా create చేస్తుంది.

```
#include<stdio.h>
int main()
{
    int *a[10],nr,nc,i,j; /* pointer array a is declared*/
```

```
    printf("Enter no of rows and no of columns: \n”);
    scanf("%d%d",&nr,&nc);
    /* Memory for each row is created and their starting addresses
are stored in the pointer array a */
    for(i=0;i<nr;i++)
    a[i]=(int*)malloc(nc*sizeof(int));
    printf("Enter the values :");
    for(i=0;i<nr;i++)
    for(j=0;j<nc;j++)
    scanf("%d", a[i]+j );
    printf(“Array Values are\n”);
    for(i=0;i<nr;i++)
     {
    for(j=0;j<nc;j++) printf("%d\t",*(a[i]+j));
    printf("\n");
    }
    return 0;
}
```

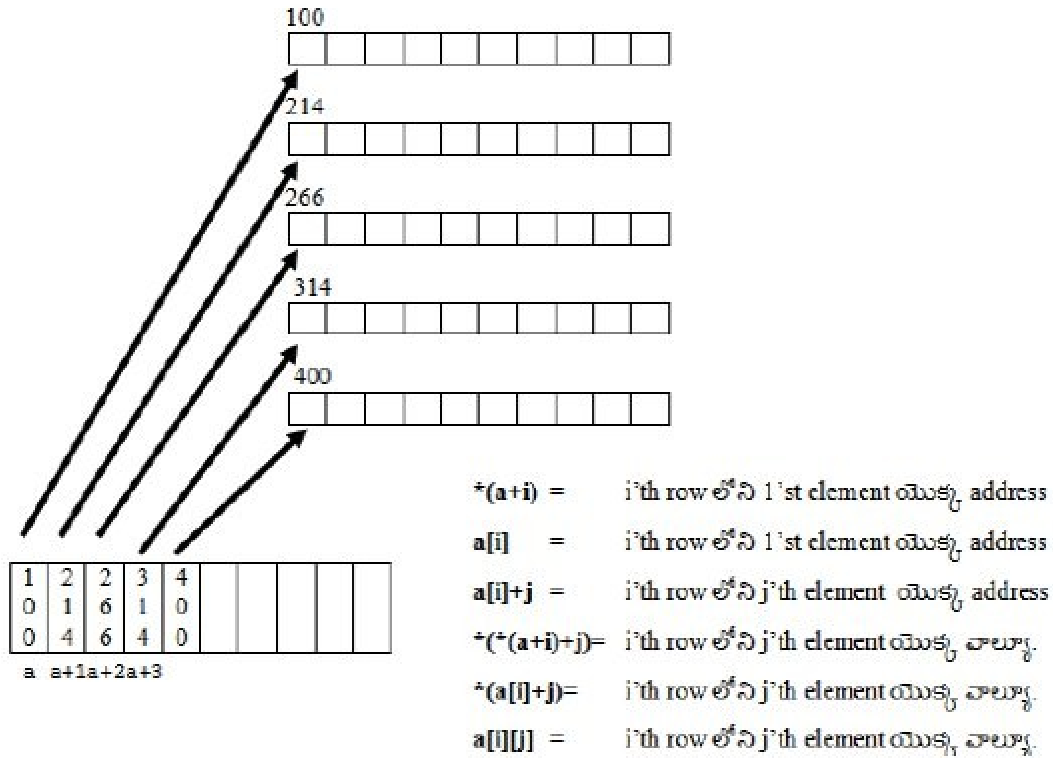
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter no of rows and no of columns:
3 4
Enter the values :2 4 4 4
4 5 5 5
5 5 9 8
Array Values are
2       4       4       4
4       5       5       5
5       5       9       8
```

15.04.4 Creating 2D Arrays Flexibly (Pointers to pointers approach)

పైన వాడిన అప్రోచ్ లో, మనం create చేయుగలిగిన 2-D arrays లో row లు మనం వాడుతున్న pointer array size కంటే ఎక్కువగా చేయలేం. అందుకని, ఈ pointer array కు కావల్సిన మెమొరీ కూడా dynamic గా allocate చేస్తే సరిపోతుంది. దీనిని pointer to pointer approach అని అంటారు. ఇక్కడ ముందుగా, ఒక pointer to pointer (int **p)ను declare చేసి, దానికి dynamic గా create చేసిన pointer array కి సంబంధించిన address ను assign చేస్తాం. ఆ తర్వాత పై example లానే జరుగుతుంది.

ఈ కింద ఇచ్చిన బొమ్మ, example program దీనిని చూపిస్తుంది.



Example 13: ఈ ప్రోగ్రాంఒక 2-D array ను pointer to pointer approach ద్వారా create చేస్తుంది.

```
#include<stdio.h>
int main()
{
    int **p,nr,nc,i,j;

    printf("Enter no of rows and no of columns: \n");
    scanf("%d%d",&nr,&nc);

    /* First a pointer array is created to store addresses of each row */

    p=(int **) malloc(nr*sizeof(int *));
    /* Memory for each row is created and their starting addresses
are stored in pointer array p */
    for(i=0;i<nr;i++)
    p[i]=(int*)malloc(nc*sizeof(int));

    printf("Enter the values :");
    for(i=0;i<nr;i++)
    for(j=0;j<nc;j++)
    scanf("%d",*(p+i)+j);
    printf(" array values are \n2-D array content is :");
    for(i=0;i<nr;i++)
    {
    for(j=0;j<nc;j++) printf("%d ",*(*(p+i)+j));
```

```
    printf("\n");
    }
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

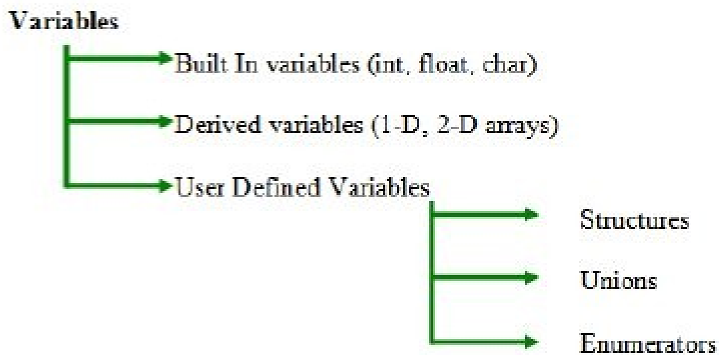
```
Enter no of rows and no of columns:
3 4
Enter the values :2 4 4 4
4 5 5 5
5 5 9 8
Array Values are
2       4       4       4
4       5       5       5
5       5       9       8
```

Structures, Unions

16. User-defined variables: Structures, Unions and Enumerators

16.1 Introduction

C లాంగ్వేజీ లో ఉన్న variables ను ఈ కింది విధంగా విభజించవచ్చు.



C లాంగ్వేజీలో 2-D arrays ఉండటంతో matrix multiplication చేయగలిగాం. అంటే matrix multiplication problem tractable (able to be done) అయింది. ఇక్కడ, array అంటే ఒకేరకమైన elements లేదా members ఉండేది. అలాగే, కొన్ని ప్రాబ్లంలను చేయడానికి మనం user defined variables ను define చేస్తాం. వీటిలోsame members లేదా వేరే టైప్ members అయినా ఉండవచ్చు . C లాంగ్వేజీలో structures, unions ,

enumerators అనే user defined variables ఉన్నాయి. వీటి గురించి ఈ పాఠంలో నేర్చుకుందాం.

9.16.2 Structures

ఒక structure ను define చేయడానికి ముందుగా దానికి ఏ పేరు, ఏ మెంబర్లను ఇవ్వాలని అనుకుంటున్నారో నిర్ణయించుకోవాలి. దానిని ఈ కింది విధంగా define చేయాలి.

```
struct struct_typename
    {
        Type member1;
        Type member2;
        –
        Type membern;
};
```

ఉదాహరణకు, DATE అనే దానిని day (d), month (m), year (y) అనే మెంబర్లతో ఇలా define చేయవచ్చు. ఆఖరున } తర్వాత ; (సెమీకోలన్) పెట్టాలి.

```
struct DATE
{
    int d;
    int m;
    int y;
};
```

ఈ కింది declaration statement ఒక DATE టైపు variable A ను, ఒక DATE type pointer variable p ను declare చేస్తుంది.

```
struct DATE A, *p;
```

వీటిని structure definition తో పాటుగా ఈ కింది విధంగా కూడా declare చేయవచ్చు.

```
struct DATE
{
    int d;
    int m;
    int y
} A, *p;
```

ఇక్కడ, A కు, మూడు integer లు d, m, y లు మెంబర్లుగా ఉన్నాయి కాబట్టి, మూడు integers కు ఎంత కావాలో అంత memory allocate అవుతుంది. అంటే, 2+2+2=6 bytes అయినా లేదా 4+4+4=12 bytes అయినా allocate అవుతుంది. అలాగే, p అనేది pointer టైప్ కాబట్టి, దానికి 4 bytes allocate అవుతాయి.

9.16.2.1 Accessing data members

A అనేది DATE టైప్ structured variable, దానిలోఉండే మెంబర్లను .(dot) అనే delimiter (separator)ను వాడి A.d, A.m, A.y లుగా చెబుతాం. ఇవి integer లు. ఒక structured variable లోపలికి డేటా రీడ్ చేసేటప్పుడు లేదా ప్రింట్ చేసేటప్పుడు మెంబరు తర్వాత మెంబరును వాడతాం. ఈ కింది ప్రోగ్రాం దీనిని విశదీకరిస్తుంది. ఇది రెండు DATE టైప్ variables A,B ల్లోకి డాటా రీడ్ చేసి ప్రింట్ చేస్తుంది.

Example 1:

```
#include<stdio.h>
struct DATE
{
    int d;
    int m;
    int y;
};
int main()
{
    struct DATE A,B;
    printf("Memory Allocated for A=%d\n", sizeof(A));
    printf("Memory Allocated for B=%d\n", sizeof(B));
    printf("Enter two Dates\n");
    scanf("%d%d%d", &A.d, &A.m, &A.y);
    scanf("%d%d%d", &B.d, &B.m, &B.y);
    printf("Given Dates are\n");
    printf("%d\t%d\t%d\n", A.d, A.m, A.y);
    printf("%d\t%d\t%d\n", B.d, B.m, B.y);
    return(0) ;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
Memory Allocated for A=12
Memory Allocated for B=12
Enter two Dates
12 09 1929
21 10 1978
Given Dates are
12      9       1929
21      10      1978
```

16.2.2 Accessing a structures data members through its pointer

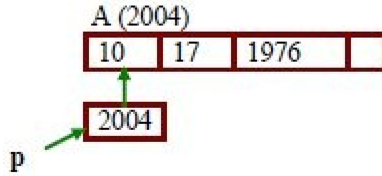
ఒక structured variable మెంబర్లను దాని pointer ద్వారా కూడా access చెయ్యవచ్చు. p అనేది ఒక DATE టైప్ structured variable (A) కు pointer అయితే అందులో (A లో) ఉండే మెంబర్లను de-referencing operator (->) ను p->d, p->m, p->y గా చెబుతాం. ఈ కింది ప్రోగ్రాం ద్వారా ఈ notation ను ఎలా ఉపయోగించాలో తెలుసుకుందాం.

Example 2:

```
#include<stdio.h>
struct DATE
{
    int d;
    int m;
    int y;
};
int main()
{
    struct DATE A, *P;
    printf("Enter a Date\n");
    scanf("%d %d %d", &A.d, &A.m, &A.y);
    P = &A; /* p is pointer to A*/
    /*All the following three ways of accessing members
    are acceptable*/
    printf("%d\t%d\t%d\n", A.d, A.m, A.y);
    printf("%d\t%d\t%d\n", P->d, P->m, P->y);
    printf("%d\t%d\t%d\n", (*P).d, (*P).m, (*P).y);
    /* Here, *p refers to A */
    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
Enter a Date
12 10 1989
12        10        1989
12        10        1989
12        10        1989
```



A కు 2004 memory allocate అయింది అనుకుంటే, p=&A అనేది రన్ అయిన తర్వాత, p కు 2004 ఇస్తుంది. ఇక్కడ, p అనేది A కు pointer, కానీ A లో ఉన్న ఏ మెంబరుకూ కాదు.

పై ప్రోగ్రాంలో, A కు memory compiler allocate చేసినది. దానికి p pointer. ఈ కింది ప్రోగ్రాంలో, dynamic గా malloc ద్వారా create చేసిన ఒక DATE టైప్ variable address ను p అనే DATE టైప్ pointer కు ఇస్తే, ఆ dynamic గా create చేసిన DATE లోని మెంబర్లను ఎలా access చేయాలో చూపిస్తుంది. ఈ కింది ప్రోగ్రాంలో వాడిన notation ను గుర్తుంచుకోవాలి.

Example 3:

```
#include<stdio.h>
struct DATE
{
    int d;
    int m;
    int y;
};
int main()
{
    struct DATE *P;
    P = (struct DATE *)malloc(sizeof(struct DATE));
    printf("Enter a Date\n");
    scanf("%d%d%d", &P->d, &P->m, &P->y);
    /* The following elements can be accessed*/
    printf("%d\t%d\t%d\n", P->d, P->m, P->y);
    printf("%d\t%d\t%d\n", (*P).d, (*P).m, (*P).y);
    return(0) ;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



9.16.2.3 Array of Structures

ఇక్కడ array of structures కూడా ఉపయోగించవచ్చు. ఉదాహరణకు, మూడు elements ఉండేటట్లు ఒక DATE టైప్ array ను ఈ కింది విధంగా dedeclare చేయవచ్చు.

struct DATE A[3];

అప్పుడు మూడు DATE టైప్ variables కు memory ఈ కింద చూపిన విధంగా ఇస్తాం.

Memory of array A.

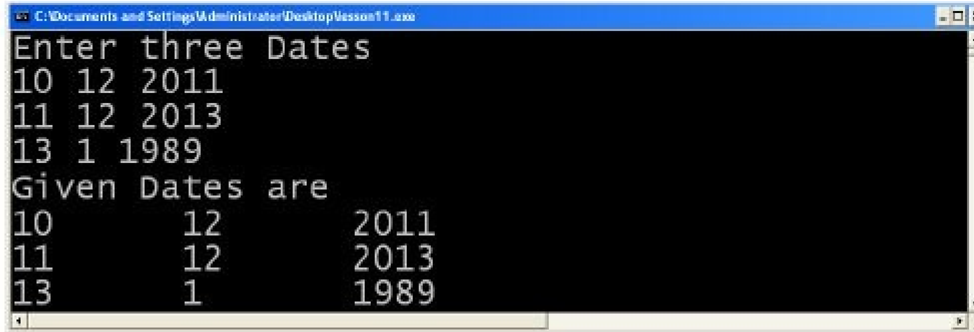
A[0]			A[1]			A[2]		

ఇప్పుడు, A[0], A[1], మరియు A[2] అనేవి DATE type variables. అలాగే, A[0].d, A[0].m, A[0].y, అనేవి A[0] లోని మెంబర్లు. ఈ కింది ప్రోగ్రాం వీటిని ఉపయోగిస్తుంది. ఈ కింది ప్రోగ్రాంలో వాడిన notation ను గుర్తుంచుకోవాలి.

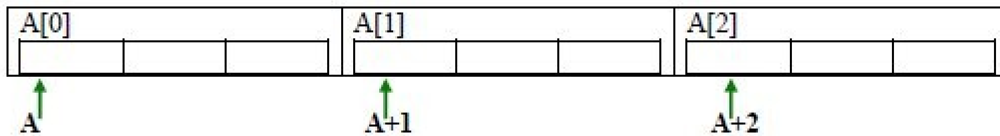
Example 4:

```
#include<stdio.h>
struct DATE
{
    int d;
    int m;
    int y;
};
int main()
{
    int i;
    struct DATE A[3]; /* A is DATE type of array */
    printf(“Enter three Dates\n”);
    for( i = 0; i < 3; i ++)
    scanf(“%d %d %d”, &A[i].d, &A[i].m, &A[i].y);
    printf(“Given Dates are\n”);
    for( i = 0; i < n; i ++)
    printf(“%d\t%d\t%d\n”, &A[i].d, &A[i].m, &A[i].y);
```

```
    return(0);
}
```
పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



Example 5: ఈ ప్రోగ్రాంలో ఒక DATE టైపు structured array (A) ను dynamic గా malloc వాడి create చేస్తున్నాం. ఈ కింది ప్రోగ్రాంలో వాడిన notation ను గుర్తుంచుకోవాలి. ఇక్కడ A అనేది array కు pointer. కాబట్టిA, A+1, A+2 లు A[0], A[1], A[2] లకు pointers. ఈ కింది పట్టికను చూడండి.




```
#include<stdio.h>
struct DATE
{
    int d;
    int m;
    int y;
};
int main()
{
    int i;
    stuct DATE *A;
    A = (struct DATE *)malloc(3*sizeof(struct DATE));
    /* Creates a structured array dynamically */
    printf("Enter Three Dates Data\n");
    for( i = 0; i < 3; i ++)
    scanf("%d%d%d", &(A+[i]->.d, &(A+[i])->.m, &A+[i]-
>.y);
```

```
    printf("Given Dates are\n");
    for( i = 0; i < 3; i ++)
    {
    /* The following three notations are equivalent*/
    printf("%d\t%d\t%d\n", &A[i].d, &A[i].m, &A[i].y);
    printf("%d\t%d\t%d\n", (A + i)->d, (A + i)->m, (A +
i)->y);
    printf("%d\t%d\t%d\n",(*(A+i)).d, (*(A+i)).m,
(*(A+i)).y);
    }
    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter three Dates
10 12 2011
11 12 2013
13 1 1989
Given Dates are
10      12      2011
11      12      2013
13      1       1989
```

16.2.4 Nested Structures

ఒక structure లోపల ఇంకో structure ను వాడవచ్చు.

```
struct STUD
{
    char name[20];
    int RNo;
    struct DATE
    {
        int d;
        int m;
        int y;
    }
    DOB;
};
```

పై ప్రోగ్రాంలో DATE definition STUD లోపల ఉంది కాబట్టి మనం DATE type ని బయట వాడలేం. కానీ పై దానినే ఈ విధంగా రాస్తే ఆ సమస్య ఉండదు.

```
struct DATE
    {
        int d;
        int m;
        int y;
    };
struct STUD
    {
        char name[20];
        int RNo;
        struct DATE DOB;
    };
```

Example 6: ఈ ప్రోగ్రాంలో STUD టైప్ variables కు సంబంధించిన notations రాశాం.

```
#include<stdio.h>
main()
{
    struct STUD A;
    printf("Enter student name, roll no, and date of
birth\n");
    scanf("%s%d%d%d%d",&A.name, &A.RNo, &A.DOB.d,
&A.DOB.m, &A.DOB.y);
    printf("%s\t%d\t%d\n%d\t%d\n", A.name, A.RNo,
A.DOB.d, A.DOB.m, A.DOB.y);
    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter student name, roll no, and date of bir
Ram 111 10 11 2012
Ram     111     10
11      2012
```

Example 7: ఈ ప్రోగ్రాంలో STUD టైప్ variables కు సంబంధించిన notations ఉన్నాయి. ఇందులో STUD టైప్ pointer ను ఎలా వాడాలో, దానికి సంబంధించిన notations గురించి తెలుసుకుందాం.

```
#include<stdio.h>
```

```
/*include the definition of above structures*/
int main()
{
    struct STUD A, *P;
    P = &A; /* P is pointer to A. Not to any member of it*/
    printf("Enter student name, roll no, and date of 
birth\n");
    scanf("%s%d%d%d%d",&A.name, &A.RNo, &A.DOB.d, 
&A.DOB.m, &A.DOB.y);
    /* The following three styles are equivalent*/
    printf("%s\t%d\t%d\n%d\t%d\n", A.name, A.RNo, 
A.DOB.d, A.DOB.m, A.DOB.y);
    printf("%s\t%d\t%d\t%d\t%d\n", P->name, P->RNo,P-
>DOB.d, P->DOB.m, P->DOB.y);
    printf("%s\t%d\t%d\n%d\t%d\n", (*P).name, 
(*P).RNo, (*P).DOB.d, (*P).DOB.m, (*P).DOB.y);
    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter student name, roll no, and date of
Rao 111 12 10 2012
Rao     111     12
10      2012
Rao     111     12      10      2012
Rao     111     12
10      2012
```

Example 8 : ఈ ప్రోగ్రాంలో STUD టైప్ pointer ఒక dynamic గా create అయిన ఒక STUD కు pointer అయితే, దానికి సంబంధించిన notations ఎలా ఉంటాయో చూద్దాం.

```
#include<stdio.h>
main()
{
    structSTRUCT STUD *P;
    P = (struct STUD *)malloc(sizeof(struct STUD));
    printf("Enter student name, roll no, and date of 
birth\n");
```

```
    scanf("%s%d%d%d%d",&P->name,&P->RNo,&P->DOB.d, &P->DOB.m,&P->DOB.y);
    printf("%s\t%d\t%d\t%d\t%d\n", P->name, P->RNo, P->DOB.d, P->DOB.m, P->DOB.y);
    printf("%s\t%d\t%d\t%d\t%d\n", (*P).name, (*P).RNo, (*P).DOB.d, (*P).DOB.m, (*P).DOB.y);
    return 0;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter student name, roll no, and date of
Rao 111 12 10 2012
Rao     111     12      10      2012
Rao     111     12      10      2012
```

16.2.5 Self Referencing Structures

పైన ఒక structure లో మరొక structure ను మెంబరుగా వాడాం. ఒక structure లోపల దాని లాంటి వాటినే members గా వాడవచ్చు. వీటిని Self Referencing Structures అని అంటారు. ఈ కింద lst అనే structure ను define చేస్తూ దానిలో next అనే lst టైప్ structured pointer ను వాడాం. ఇలాంటి వాటిని data structures లో ఎక్కువగా ఉపయోగిస్తాం.

```
struct lst
{
    int n;
    struct lst *next;
};
```

Example 9: ఈ ప్రోగ్రాంలో ఒక simple linked list ఎలా create చేయాలో చూద్దాం.

```
#include<stdio.h>
struct lst
{
    int n;
    struct lst *next;
};
int main()
{
    struct lst A,B,C, *H;
```

```
    A.n=100;
    /* A.n is an integer thus we are assigning 100.*/
    B.n=190;
    /* B.n is an integer thus we are assigning 190.*/
    C.n=11;
    /* C.n is an integer thus we are assigning 11.*/
    A.next=&B;
    /* We know A.next is a lst type of pointer and we are
assigning address of a lst type of object (B). Thus, no
violation of rules. */
    B.next=&C;
    /* We also know B.next is a lst type of pointer and we
are assigning address of a lst type of object (C). Thus, no
violation of rules. */
    C.next=0;
    /* We know A.next is a lst type of pointer and we are
assigning 0. One can assign zero a a pointer. Thus, no
violation of rules. */
    H=&A;
    /* We know H is a lst type of pointer and we are
assigning address of a lst type of object (A). Thus, no
violation of rules. */
    printf(Elements il the linked list are\n");
    while(H)
    {
    printf("%d\n", H->n);
    H=H->next;
    }
    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Elements in the linked list are
100
190
11
```

9.16.2.6 Initializing Data Members of a Structure

మనం structured variables మెంబర్లను initialize కూడా చేయవచ్చు. ఈ కింది ప్రోగ్రాం దానిని వివరిస్తుంది. ఉదాహరణకు, DATE type structured variable ను ఇలా initialize చేయవచ్చు.

struct DATE A={10,12,2009}; అప్పుడు, A.d=10, A.m=12, A.y=2009 అవుతాయి.

Example 10:

```
#include<stdio.h>
struct DATE
{
    int d;
    int m;
    int y;
};
int main()
{
    struct DATE A={10,12,2009},B[2]={{10,12,2008},
{2,2,2009}},C[]= {{10,12,2008},{2,2,2009}};

    printf("%d\t%d\t%d\n", A.d, A.m, A.y);
    printf("%d\t%d\t%d\n", B[0].d, B[0].m, B[0].y);
    printf("%d\t%d\t%d\n", C[1].d, C[1].m, C[1].y);
    return(0) ;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

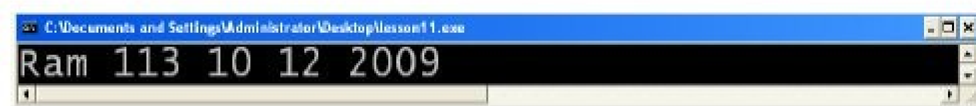


Example 11: nested structured variables మెంబర్లను initialize చేసే ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
struct DATE
    {
    int d;
    int m;
```

```
    int y;
    };
struct STUD
    {
    char name[20];
    int RNo;
    struct DATE DOB;
    };
int main()
{
    struct STUD A={"Ram", 113,10,12,2009};
    printf("%s %d %d %d %d\n", A.name, A.RNo,
A.DOB.d, A.DOB.m, A.DOB.y);
    return(0) ;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



16.2.7 Passing structures to functions

మనం structured variables ను, structured pointer variables ను functions లోపలికి పంపించవచ్చు. ఈ కింది declaration లో A అనేది passing by value స్టైల్, p అనేది passing by address స్టైల్.

```
void xyz(struct DATE A, struct DATE *p);
```

Example 12: Function, PRINTDATE() అనేది ఒక DATE టైప్ variable ను argument గా తీసుకొని దాని మెంబర్ల విలువలను ప్రింట్ చేస్తుంది. READDATE() function ఒక DATE టైప్ variable address తీసుకుని దాని లోపలికి data రీడ్ చేస్తుంది.

```
#include<stdio.h>
struct DATE
    {
    int d;
    int m;
    int y;
    };
void PRINTDATE(struct DATE A)
    {
```

```
        printf("%d\t%d\t%d\n", A.d, A.m, A.y);
    }
void READDATE(struct DATE *A)
     {
        scanf ("%d%d%d", &A->d,&A-> m, &A->y);
    }
int main()
{
    struct DATE X;
    printf("Enter Date value\n");
    READDATE(&X);
    printf("Data Given is\n");
    PRINTDATE(X);
    return(0) ;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
Enter Date value
10 12 2012
Data Given is
10        12        2012
```

Example 13 : ఈ ప్రోగ్రాంలో PRINTDATE(), READDATE(), functions ను వాడి structured DATE array లోపలికి, dynamic structured DATE array లోపలికి డేటా ఎలా రీడ్ చేసి ఎలా ప్రింట్ చేయాలో చూపిస్తుంది.

```
#include<stdio.h>
int main()
    {
    struct STRUCT DATE A[3], *X;,
    int i;
    printf("Enter Three Dates \n");
    for( i = 0; i < 3; i ++) READDATE(&A[i]);
    printf("Given Dates are\n");
    printf("Given Dates are\n");
    for( i = 0; i < 3; i ++) PRINTDATE(A[i]);
```

```
    X = (struct DATE *)malloc(3*sizeof(struct DATE));
    printf("Enter Three Dates \n");
    for( i = 0; i < 3; i ++) READDATE(X + i);
    printf("Given Dates are\n");
    for( i = 0; i < 3; i ++) PRINTDATE(X[i]);
    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter three dates
10 12 2001
12 10 2001
12 12 2012
Given dates are
10        12        2001
12        10        2001
12        12        2012
Enter three dates again
10 12 2001
12 10 2002
12 08 2010
Given dates are
10        12        2001
12        10        2002
12        8         2010
```

9.16.2.8 Returning structures from functions

మనం structures ను, structured addresses ను functions నుంచి రిటర్న్ చేయవచ్చు. ఈ కింది ప్రోగ్రాం దీనిని explain చేస్తుంది. ఇది DATE టైప్ structured variable ని function నుంచి రిటర్న్ చేస్తుంది.

Example 14:

```
#include<stdio.h>
struct DATE XYZ()
    {
    struct DATE A;
    printf("Enter a Value for Date\n");
    scanf("%d%d%d", &A.d, &A.m, &A.y);
    return A;
}
int main()
{
```

```
    struct DATE B;
    B=XYZ();
    printf(" Date read in the function = %d\t%d\t%d\n",
B.d, B.m, B.y);
    return(0) ;
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.



16.3 Unions

ఇది కూడా ఒక user defined type. చాలా తక్కువగా అవసరమవుతుంది. దీనిలో చాలా మెంబర్లు ఉన్నా, ఏదో ఒకటి మాత్రమే active లేదా meaningful గా ఉంటుంది.

ఉదాహరణకు ఒక warehouse manager, తన warehouse లో ఉన్న items వాటి LR number లేదా description తో నిర్వహిస్తాడు. అంటే ఏదో ఒకటే ఉంటుంది. అలాంటప్పుడు items ను define చేయడానికి Union వాడతాం.

```
union Item
{
    int LRNo;
    char desc[20];
};
```

Example 15: ఈ కింది ప్రోగ్రాంతో union Item ను ఎలా ఉపయోగించాలో తెలుసుకుందాం.

```
#include<stdio.h>
union Item
{
    int LRNo;
    char desc[20];
};
int main()
{
    union Item A,B;
    printf("Enter LRNo and Description\n");
    scanf("%d%s", &A.LRNo, B.desc);
    printf("Correct way of accessing\n");
```

```
    printf("LRNo=%d Description %s\n", A.LRNo, B.desc);
    printf("In-Correct way of accessing\n");
    printf("LRNo=%s Description %d\n", A.desc, B.LRNo);

    return(0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Enter LRNo and Description
111 Fridge
Correct way of accessing
LRNo=111 Description Fridge
In-Correct way of accessing
LRNo=o Description 1684632134
```

9.16.4 Enumerations

వీటి ద్వారా symbolic constants ను define చేయవచ్చు. ఈ కింద ఇచ్చిన దాని ద్వారా VAL1, VAL2, అనే symbolic constants define అవుతాయి, వాటి విలువలు 0,1...లాగా ఉంటాయి.

```
enum enum_name
    {
        VAL1,
        VAL2,
        .....
        VAL9
    };
```

ఈ symbolic constants మనకు కావలసిన విలువలు ఇస్తూ కూడా enumerator ను define చేయవచ్చు.

```
enum enum_name
{
    VAL1=1,
    VAL2=70,
    VAL3,
    VAL4=90,
    .....
    VAL9
};
```

ఇప్పుడు ఈ కింది దాని ద్వారా TRUE, FALSE symbolic constants ను define చేస్తున్నాం.

```
enum
{
    FALSE,
    TRUE
};
```

ఈ కింది ప్రోగ్రాం segment లో వీటిని వాడుతున్నాం.

```
switch(answer)
{
    case FALSE: ----
        ---
        break;
    case TRUE: __
        ----
        break;
}
```

ఈ కింది దానితో పోలిస్తే ఇది కొద్దిగా సులభంగా అర్ధమవుతుంది నిజానికి రెండూ ఒకే పని చేస్తాయి.

```
switch(answer)
{
    case 0: ----
    ---
    break;
    case 1: __
    ----
    break;
}
```

16.5 Typedef

ప్రోగ్రాంల రీడబిలిటిని పెంచేందుకు, typedef ద్వారా కొత్త పేర్లు int, float, double టైప్ వాటికి బదులుగా వాడవచ్చు. ఉదాహరణకి, ఈ కింద int కు వేరే పేరు LENGTH అని ఇస్తుంది.

```
typedef int LENGTH;
```

ఇప్పుడు కింద ఇచ్చిన రెండు statements లోనూ length, width, height అనేవి int టైపే. కానీ మొదటి statement కొద్దిగా చదవడానికి బాగుంటుంది.

```
LENGTH length, width, height;
int length, width, height;
```

అదే విధంగా మనం define చేసిన STUD structure కు, ఈ కింది విధంగా వేరే పేరు ఇవ్వవచ్చు.

```
typedef struct STUD PEOPLE;
```

ఇప్పుడు ఈ కింద ఇచ్చిన రెండూ ఒకటే.

```
struct STUD A;
PEOPLE A;
```

డేటాని ఫైల్ లోకి రాయడం ఎలా?

17. డేటాని ఫైల్ లోకి రాయడం ఎలా?

17.1 Introduction

ఇప్పటి వరకు మనం రాసిన అన్ని ప్రోగ్రాంలు కావల్సిన దానిని keyboard నుంచి తీసుకొని ఫలితాన్ని స్క్రీను మీద చూపించాయి. ఈ పాఠంలో ఒక file నుంచి డేటా రీడ్ చేసి, ఫలితాన్నిfile లోకి రాయడం ఎలాగో తెలుసుకుందాం. దీనికి మనం FILE * టైప్ variables ను ఉపయోగించాలి. వీటిని చేసేందుకు చాలా readymade function లు ఉన్నాయి. అందులో కొన్నింటి గురించి తెలుసుకుందాం.

fopen

ఒక file ను ఓపెన్ చేయడానికి, fopen అనే function ను ఉపయోగిస్తాం. దానికి file పేరు (దాని PATH తో కలిసి), ఎందుకు ఆ file ను ఓపెన్ చేద్దామనుకుంటున్నాం అనేవి ఈ కింది విధంగా రెండు string టైప్ argument లను పంపుతూ చెప్పాలి. రెండో దానిని mode string అని అంటారు. ఈ function fail అయితే 0 రిటర్న్ చేస్తుంది.

```
FILE* fopen( char *path, char *mode)
```

ఉదాహరణకు, “results” అనే file ను డేటా రాయడం కోసం ఓపెన్ చేయాలంటే, ఈ కింద ఇచ్చిన విధంగా ఒక FILE* టైప్ variable ను declare చేసి, fopen function ను call చేయాలి.

```
FILE *p2;
p2=fopen(“results”,”w”);
```

ఈ కింద ఇచ్చిన టేబుల్, ఏ mode file ఓపెన్ చేయాలంటే, ఏ mode string ఉపయోగించాలో తెలుపుతుంది.

Mode of Opening	Remarks
R	ఉన్న ఫైల్ ను రీడ్ చేయడానికి
r+	ఉన్న ఫైల్ ను రీడ్ చేయడానికి ,రాయడానికి
w	ఒక ఫైల్ ను రాయడానికి దీనిని వాడతాం. మనం ఇచ్చిన పేరుతో ఫైల్ లేకపోతే కొత్త ఫైల్ ఓపెన్ అవుతుంది, ఉంటే దానిలో ఉండే matter పోతుంది.
w+	ఒక ఫైల్ ను రీడింగుకు, రాయడానికి దీనిని ఉపయోగిస్తాం. మనం ఇచ్చిన పేరుతో ఫైల్ లేకపోతే కొత్త ఫైల్ ఓపెన్ అవుతుంది, ఉంటే దానిలో ఉండే matter పోతుంది.
a	ఒక ఫైల్ ను append చేయడానికి దీనిని ఉపయోగిస్తాం. మనం ఇచ్చిన పేరుతో ఫైల్ లేకపోతే కొత్త ఫైల్ ఓపెన్ అవుతుంది, ఉంటే దానిలో ఉండే దానికి మనం రాసేది చివర లో append అవుతుంది.
a+	ఒక ఫైల్ ను రీడింగు append చేయడానికి దీనిని ఉపయోగిస్తాం. మనం ఇచ్చిన పేరుతో ఫైల్ లేకపోతే కొత్త ఫైల్ ఓపెన్ అవుతుంది, ఉంటే దానిలో ఉండే దానికి మనం రాసేది చివర లో append అవుతుంది.

The fprintf() & fscanf() functions

వీటిని ఉపయోగించి ఫైల్ నుంచి డేటా రీడ్ లేదా ప్రింట్ చేయవచ్చు. printf, scanf లతో పోలిస్తే, వీటికి first argument గా మనం రీడ్/రాయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer ఇవ్వాలి.

వీటి declarations ఈ కింద విధంగా ఉంటాయి.

fprintf(fp,"format string", list of arguments to be printed);

fscanf(fp,"format string", addresses of arguments into which data has to be read);

Other I/O Functions

int fgetc(FILE *ip);

ఈ function కు first argument గా రీడ్ చేయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది ఒక character ను దాంట్లోనుంచి రీడ్ చేసి రిటర్న్ చేస్తుంది.

int fputc(char V, FILE *ip);

ఈ function కు arguments గా ఒక character, మనం రాయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది ఆ character ను ఫైల్ లో రాస్తుంది.

char *fgets(char *s, int n, FILE *ip);

ఈ function కు argumenst గా ఒక string variable address (name), ఒక integer, రీడ్ చేయాలనుకుంటున్న ఫైల్ కి సంబంధించిన FILE

pointer ఇస్తే అది ఒక string ను దాంట్లోనుంచి రీడ్ చేసి ఇచ్చిన string లో పెడుతుంది.

fputs(char *s, FILE *ip);

ఈ function కు arguments గా ఒక string, మనం రాయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది ఆ string ను ఫైల్ లో రాస్తుంది.

The fwrite(), fread() functions

ఇవి binary mode లో file మీద I/O చేయడానికి వాడతారు. ఇంతకు ముందు చర్చించిన functions I/O చేస్తున్నప్పుడు ASCII form లో చేస్తాయి. అదే ఈ functions అయితే, binary pattern ను ditto లాగా I/O లో తీసుకుంటాయి.

```
int fread( void *ptr, size_t sz, size_t n, FILE *ftr);
int fwrite( void *ptr, size_t sz, size_t n, FILE *ftr);
```

ఈ fread function sz bytes ఉండే n elements ను, ఫైల్ pointer ftr point చేస్తున్న ఫైల్ లోనుంచి ptr point చేస్తున్న memory లో పెడుతుంది.
ఈ fwrite function sz bytes ఉండే n elements ను, ఫైల్ pointer ftr point చేస్తున్న ఫైల్ లోపలికి ptr point చేస్తున్న memory లో నుంచి కాపీ చేస్తుంది.

A Note on Formatted/Un-formatted I/O

మనం fprintf(), fputs(), లతో రాస్తే వచ్చే file ను text file టైప్, దానిని notepad, wordpad వంటి editors సహాయంతో చూడొచ్చు, ఎడిట్ చేయొచ్చు. వీటిద్వారా ఫైల్ లోకి ఏం రాసినా వాటిని ASCII లోకి మార్చి రాస్తాయి. ఉదాహరణకు, ఒక integer variable, x=32113, అనేది memory లో (RAM లో) 2 bytes లో ఉంటుంది. దానిని fprintf() ద్వారా ఫైల్ లోపలికి రాస్తే 5 bytes తీసుకుంటుంది. అదే fwrite ద్వారా రాస్తే ఫైల్ లోకూడా 2 bytes మాత్రమే తీసుకుంటుంది. fwrite ద్వారా రాసిన ఫైల్ లను un-formatted or binary ఫైల్ లు అని అంటారు. ఇవి తక్కువ స్పేస్ తీసుకుంటాయి కాబట్టి I/O వేగవంతమవుతుంది.

Closing a file

int fclose(FILE *stream)

ఈ function కు argument గా మనం క్లోజ్ చేయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది ఆ ఫైల్ ను క్లోజ్ చేస్తుంది.

int fflush(FILE *stream)

ఈ function కు argument గా మనం flush చేయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది ఆ ఫైల్ వాడుతున్న అన్ని buffers ను flush చేస్తుంది.

Stream Posistioning

void rewind(FILE *stream);

ఈ function కు argument లాగా మనం రీడ్/రాయాలని అనుకుంటున్న ఫైల్ కి

సంబంధించిన FILE pointer ఇస్తే అది దానిలో ఉండే file కి సంబంధించిన invisible pointer ను file begining కు తెస్తుంది.

fseek(file pointer,offset, position);

దీని ద్వారా ఫైల్ లో invisible pointer ను ఎక్కడ కావాలంటే అక్కడ పెట్టవచ్చు. దీనికి మొదటి argument మనం సీక్ చేయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer, రెండోది ఒక integer, మూడోది ఎక్కడ నుంచి పొజిషన్ ను చెపుతున్నాం. దీని గురించి కింద ఏ విలువ ఇస్తే ఏం వస్తుందో ఇచ్చాం.

Value Meaning
0 Beginning of the file
1 Current position
2 End of the file.

int ftell(FILE*);

ఈ function కు first argument గా రీడ్ చేయాలని అనుకుంటున్న ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది ఒక long నంబరును రిటర్న్ చేస్తుంది. ఇది ఆ ఫైల్ లో రీడింగ్ pointer, beginning నుంచి ఎన్ని bytes దూరంలో ఉందో ఇస్తుంది.

int feof(FILE *);

ఈ function కు first argument గా ఓపెన్ చేసిఉన్న ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది ఫైల్ pointer చివరలో ఉంటే 1 లేకపోతే 0 రిటర్న్ చేస్తుంది.

int ferror(FILE *);

ఈ function కు first argument గా రీడ్/రాసిన ఫైల్ కి సంబంధించిన FILE pointer ఇస్తే అది 1 లేక 0 రిటర్న్ చేస్తుంది. ఆ ఫైల్ మీద చేసిన ఏదైనా ఆపరేషను error అయి ఉంటే 1 లేకపోతే 0 రిటర్న్ అవుతుంది.

Example 1: ఈ కింది ప్రోగ్రాం fopen, fprintf, scanf, fclose, fflush functions ను ఎలా ఉపయోగించాలో చూపిస్తుంది. ఇది మూడు integers ను keyboard నుంచి రీడ్ చేసి TEST అనే file ను open చేసి దానిలోనికి రాస్తుంది. ఆ ఫైల్ ను క్లోజ్ చేసి మళ్లీ ఓపెన్ చేసి దానిలో ఉండే మూడు విలువలను రీడ్ చేసి (వేరే variables లోకి) స్క్రీన్ మీద చూపిస్తుంది..

```
#include<stdio.h>
int main()
{
    int x,y,z, p,q,r;
    FILE *IPP;
    printf("Enter Three Integers\n");
    scanf("%d%d%d", &x, &y, &z);
    IPP=fopen("TEST", "w");
```

```
    fprintf(IPP,"%d\n%d\n%d\n", x,y,z);
    fflush(IPP); /* Not always needed*/
    fclose(IPP);
    IPP=fopen("TEST", "r"); /*re-opened the file but in
reading mode */
    fscanf(IPP,"%d%d%d", &p, &q, &r);
    printf("Numbers read from the file are=%d %d %d\n",
p,q,r);
    return 0;
}
```

Example 2: ఈ ప్రోగ్రాం ఒక ఫైల్ ను ఓపెన్ చేసి దాని content ను line number తో కలిపి ప్రింట్ చేస్తుంది. ఇక్కడ fgets ను ఉపయోగించి ఫైల్ నుంచి ఒక్కో లైన్ రీడ్ చేసి , ఒక కౌంటరు విలువను ప్రింట్ చేస్తున్నాం. లైను రీడ్ చేసినప్పుడల్లా కౌంటరు విలువను ఒకటి పెంచుతున్నాం.

```
#include <stdio.h>
#define LINE_LENGTH 512
int main(int argc, char *argv[])
{
    FILE* fpp;
    char line[LINE_LENGTH];
    int count=0;
    fpp=fopen(argv[1],"r");
    /* Count up the lines here. */
    while ( fgets(line, LINE_LENGTH, fpp) != NULL)
    {
    printf("%3d %s\n", count++. Line);
    }
    fclose(fpp);
    return 0;
}
```

Example 3: ఈ ప్రోగ్రాం file name ని command line ద్వారా తీసుకుని దాని size ను ప్రింట్ చేస్తుంది. ఫైల్ ని ఓపెన్ చేసిన తర్వాత fseek() వాడి file pointer ను end of the file, అంటే చివర లో పెట్టి అప్పుడు ftell() call చేస్తే file size వస్తుంది.

```
#include<stdio.h>
int main(int argc, char*argv[])
{
    FILE *ipp;
```

```
    if(argc!=2)
    {
    printf("Only one argument is needed\n");
    exit(-1);
    }
    ipp=fopen(argv[1], "r");
    if(ipp==0) {
    printf("Error in opening file");
    exit(-1);
    }
    fseek(ipp,0,2); /* positioning the reading pointer at
EOF*/
    printf("File Size=%ld\n", ftell(ipp)); /* Calling ftell()
function*/
    fclose(ipp);
    return 0;
}
```

Example 4: ఈ ప్రోగ్రాం operating system లో ఉండే COPY command లా పనిచేయడనికి ఉద్దేశించినది. అంటే, ఇది source file name ను first command line argument గా , duplicate file name ను second command line argument గా తీసుకొని source file లో ఉండేదానిని destination file లోపలికి కాపీ చేస్తుంది. ఇక్కడ fgetc ని ఉపయోగించి character తర్వాత character ను రీడ్ చేసి fputc తో EOF వచ్చేంత వరకు చేస్తాం.

```
#include<stdio.h>
int main(int argc, char*argv[])
{
    FILE *ipp, *opp;
    if(argc!=3)
    {
    printf("Two arguments are needed\n");
    exit(-1);
    }
    ipp=fopen(argv[1], "r");
    if(ipp==0)
    {
    printf("Error in opening file");
    exit(-1);
```

```
    }
    opp=fopen(argv[2],"w");
    while(!feof(ipp))
    fputc(fgetc(ipp),opp);
    fclose(ipp);
    fclose(opp);
    return (0);
}
```

Example 5: ఈ example ను Formatted I/O,binary I/O ల మధ్య ఉండే తేడాను చూపేందుకు తీసుకున్నాం. ఇక్కడ, ఒక integer array ఉన్న విలువలను fprintf() ద్వారా ఒక ఫైల్ లోపలికి, fwrite() ద్వారా ఇంకొక ఫైల్ లోపలికి రాశాం. ప్రోగ్రామ్ పూర్తయిన తర్వాత రెండింటిని notepad వంటి editor ద్వారా ఓపెన్ చేస్తే fwrite() ద్వారా రాసిన ఫైల్ లో ఉండేది చూడలేం. ఎందుకంటే, అది binary ఫైల్. అంతేకాకుండా, ఈ binary ఫైల్ ను ఓపెన్ చేసి అందులో ఉన్న విలువలను fread ద్వారా రీడ్ చేసి ప్రింట్ చేస్తుంది.

```
#include<stdio.h>
int main()
{
    int x[5]={31111,31100,31001,31011,31000}, N,i;
    FILE *IPP, *OPP;
    printf("Writing into file using Formatted manner\n");
    IPP=fopen("TEST1", "w");
    for(i=0;i<5;i++)fprintf(IPP,"%d\n", x[i]);
    fflush(IPP);
    fclose(IPP);
    printf("Writing into file using Un-Formatted(Binary)
manner\n");
    OPP=fopen("TEST2", "w+");
    fwrite(x, sizeof(int), 5, OPP);
    fflush(OPP);
    rewind(OPP);
    printf("Reading Numbers From Binary File\n");
    for(i=0;i<5;i++)
     {
    fread(&N, sizeof(int),1,OPP);
    printf("%d\n", N);
    }
```

```
    return 0;
}
```

Example 6: ఈ ప్రోగ్రాం ఒక input file ("student.dat") ఓపెన్ చేయాలి. అందులో ఎంతమంది విద్యార్థులు ఉన్నారనేది మొదటి లైనులో ఉంటుంది. తర్వాత, ఒక్కో లైనులో ఒక్కో విద్యార్థి ఐదు test మార్కులు ఈ కింది విధంగా ఉంటాయి.

```
5
90    20    20    20    90
90    99    90    99    89
43    33    33    22    29
44    40    40    40    40
90    90    90    90    90
```

ఈ ప్రోగ్రాం మార్కులు, గ్రేడ్స్ ఈ కింది విధంగా ఇవ్వాలి. ఇంతకు ముందు చాప్టరులలో గ్రేడింగ్ రూల్స్, పాయింట్స్ గురించి చర్చించాం. ఇందులో కూడా వాటినే ఉపయోగించాం.

```
90 A    20 D    20 D    20 D    90 A    6.400000
90 A    99 A    90 A    99 A    89 A    10.000000
43 C    33 D    33 D    22 D    29 D    4.400000
44 C    40 C    40 C    40 C    40 C    6.000000
90 A    90 A    90 A    90 A    90 A    10.000000
```

```
#include<stdio.h>
char Grade(int n)
    {
    return('A' + (4 - n/20) +( n ==100));
    }
    int Points(char g)
    {
    return(2 * ( 70-g) );
    }
int main()
{
    int a[10][5], n, i, j;
    char v;
    float s;
    FILE *IPP;
    IPP=fopen("student.dat", "r");
    fscanf(IPP,"%d", &n);
    printf("Reading Students Marks in five tests\n");
    for(i = 0; i < n; i++)
```

```
for(j = 0; j < 5; j++)
fscanf(IPP,"%d", &a[i][j]);
for(i = 0; i < n; i++){
s = 0;
for(j = 0; j < 5; j++){
v = Grade(a[i][j]);
/* Here, a[i][j] is ith student jth test marks. Which is integer*/
s += Points(v);
printf("%d %c\t", a[i][j], v);
}
printf("%f\n", s/5);
}
fclose(IPP);
return (0);
}
```

పైన ఇచ్చిన ప్రోగ్రాంను రన్ చేసినప్పుడు కంప్యూటరు మీద ఈ విధంగా ఉంటుంది.

```
C:\Documents and Settings\Administrator\Desktop\lesson11.exe
Reading Students Marks in five tests
90 A    20 D    20 D    20 D    90 A    6.40
90 A    99 A    90 A    99 A    89 A    10.0
43 C    33 D    33 D    22 D    29 D    4.40
44 C    40 C    40 C    40 C    40 C    6.00
90 A    90 A    90 A    90 A    90 A    10.0
```

Example 7: ఈ ప్రోగ్రాం కొన్ని STUD అనే structure టైప్ variables information ను binary mode లోఒక ఫైల్ లోకి రాసి మళ్లీ రీడ్ చేసి స్క్రీన్ మీద ప్రింట్ చేస్తుంది. అంతేకాకుండా, ఇలాంటి ఫైల్ మీద random accessing ఎలా చేయాలో కూడా చూపిస్తుంది.

```
#include<stdio.h>
struct DATE
{
int d;
int m;
int y;
};
struct STUD
{
```

```
    char name[20];
    int RNo;
    struct DATE DOB;
    };
    void READSTUD(struct STUD *A)
    {
    scanf("%s%d%d%d%d", A->name, &A->RNo, &A-
>DOB.d, &A->DOB.m, &A->DOB.y);
    }

    void PRINTSTUD(struct STUD *A)
    {
    printf("%s\t%d\t%d\t%d\t%d\n", A->name, A->RNo,
A->DOB.d, A->DOB.m, A->DOB.y);
    }
int main()
    {
    struct STUD STD[5],X;
    int i,n=5,p;
    FILE *OPP;
    for(i=0;i<n;i++){ printf("Enter Next Stud Name, Roll
No and Date of Birth\n");
    READSTUD(&STD[i]);
    }
    OPP=fopen("stud.dat", "w");
    fwrite(STD, sizeof(struct STUD), 5, OPP);
    fclose(OPP);

    OPP=fopen("stud.dat", "r");
    printf("Students Details Read from the file\n");
    for(i=0;i<n;i++){
    fread(&X, sizeof(struct STUD),1,OPP);
    PRINTSTUD(&X);
    }

    printf("Now we are doing random accessing on the
file\n");
    printf("Enter Which Student Details We want\n");
```

```
    scanf("%d", &p);

    rewind(OPP); /* rewinding the file*/
    fseek(OPP, sizeof(struct STUD)*(p-1), 0);
    fread(&X, sizeof(struct STUD),1,OPP);
    PRINTSTUD(&X);
    fclose(OPP);
    return 0;
}
```